Principles for Entity Authentication *

Michele Bugliesi, Riccardo Focardi, Matteo Maffei, and Fabio Tudone

Dipartimento di Informatica
Università Ca' Foscari, Venezia, Italy

Technical Report CS-2002-16

Abstract. We study the roles of message components in authentication protocols. In particular, we investigate how a certain component contributes to the task of achieving entity authentication. To this aim, we isolate a minimal set of roles that enables us to extract general principles that should be followed to avoid attacks. We then formalize these principles in terms of rules for protocol parties and we prove that any protocol following these rules (i.e., designed according to the principles) will achieve entity authentication.

1 Introduction

Security protocols, also known as cryptographic protocols, are designed to reach specific security goals in (possibly) completely hostile environments, like, e.g., Internet. Typical security goals include the communication of a secret between two trusted entities, an authenticated message exchange, the generation and the sharing of a session key, the authentication of an entity with respect to another entity (e.g., to a server), and more.

The design of security protocols is complex and often error prone, as witnessed by the many attacks to long standing protocols reported in the recent literature on the subject (see, e.g., [9, 10, 13, 18–20]). Most of these attacks do not need to break cryptography to be performed. In fact, even when cryptography is assumed as a fully reliable building-block, an intruder can engage a number of potentially dangerous actions: it can intercept messages before they reach their destination, insert new messages, read those that travel along the communication links and forge new ones using the knowledge it has previously gained. All these actions are at disposal for an intruder to try to break a protocol by exploiting a flaw in the underlying protocol logic.

In this paper, we focus on (shared-key based) cryptographic protocols for *entity authentication*, i.e. on protocols that enable one entity (i.e., the *claimant*) to prove its claimed identity to another entity (i.e., the *verifier*)[14, 5], and discuss a novel methodology to protect such protocols from attacks to their logic.

A typical source of flaws in security protocols, and specifically in authentication protocols, is a poor interpretation of messages: sometimes it is assumed that a certain message provides more guarantees than it really does. As observed in [4] "every message should say what it means, i.e., its interpretation should depend only on its content". As a trivial example consider the protocol:

$$A \to B : \{\text{“Here I am!”}\}_{K_{AB}}$$

in which *Alice* is sending to *Bob* the message "Here I am!" encrypted with a long-term key shared between them. When Bob receives the message, he could erroneously deduce that it has been generated by Alice since it is encrypted with a key that only Alice (besides Bob himself) knows. However, this is not true if Bob previously run the same protocol with Alice (with exchanged roles). In that case, the following, well-known, *reflection attack* could be exploited by an intruder E:

* This work has been partially supported by the MURST project 'Modelli formali per la sicurezza" and the EU Contract IST-2001-32617 'Models and Types for Security in Mobile Distributed Systems" (MyThS).

$$a) \quad B \to E(A) : \{\text{“Here I am!”}\}_{K_{AB}}$$
$$b) \quad E(A) \to B : \{\text{“Here I am!”}\}_{K_{AB}}$$

Here E is pretending to be A (denoted with $E(A)$) in the first run a, intercepting the message sent by Bob, and is replaying the same message back to Bob in a second run b. As a consequence, B erroneously interprets his own message as sent by Alice. The problem is that Bob is assuming that the message has been generated by Alice but this is not explicitly indicated in the message

Starting from this need to make the interpretation of messages unambiguous, we study in detail the roles of message components in authentication protocols. In particular, we investigate how a certain component, with a given role, contributes to the task of achieving entity authentication. As noticed above, a way to avoid reflection attacks is to make the identifier of the message sender explicit:

$$A \to B : \{\text{“Here is }\mathbf{A}\text{!”}\}_{K_{AB}}$$

To motivate why adding “A” prevents the flaw, it is useful to investigate which role is played by “A” in the authentication step. A *low-level* approach could lead to state that “A represents the entity which encrypted the message”. When Bob receives the message, he is ensured that he was not the creator, as A plays the role of “encrypter”. This interpretation is fine for this simple example, but it is unpractical for more complicate authentication protocols as it only conveys information about how encryption is used. Furthermore, is not always the case that encrypter identifiers are made explicit in protocol messages.

We propose a refined, and more abstract, approach whereby we assign to entity identifiers the roles entities typically play in authentication protocols. As an example, in the message above, we say that “A represents the *claimant* of the authentication protocol”. This role is, in a sense, more abstract than the role “encrypter” since it does not directly relate to who encrypted what. When B receives the message, he can deduce that “A is the claimant of the protocol”, and avoid reflection attacks based on this more abstract information.

We consider three fundamental roles for entity identifiers: (i) *Claimant*, an identifier representing a principal willing to authenticate itself to some other principal; (ii) *Intended verifier*: an identifier representing a principal that a claimant wants to act as a verifier for its response; (iii) *Key owner*: an identifier representing a principal that owns a session key. We decided to choose these three roles after studying many existing shared-key authentication protocols. In particular, we noticed that every message encrypted with a long-term key contains an identifier that can be interpreted as playing one such role.

There are other message components that are necessary to make these roles effective during the authentication task. In particular, each role needs a time-dependent element, like e.g., nonces, timestamps, sequence numbers, that provides *freshness*. Here we only consider *nonces*, i.e., numbers that are used only once: they are typically generated as challenges, and sent back inside encrypted messages, to be *checked* and thus prove that a certain message is not a replay of an old one. Together with nonces, we associate entities with the session key they possess (this is always true of entities playing the *Key owner* role). This is useful for distribution of session keys, where a trusted server wants to communicate that a certain entity owns a certain session key.

We cast all of these ideas into the notion of *authentication block*, i.e., an abstraction of an encrypted message which only contains the message components that are directly relevant to authentication, plus their role in the authentication task. Authentication blocks have the form:

$$\langle \mathsf{R}, I, N, K_s \rangle_K$$

where R is one of the three roles above, I represents an entity playing that role, N is a nonce, and K_s, if present, is a session key owned by I. An entity which decrypts an authentication block, checking the nonce N, gives the following unique interpretation to the block components: "Entity I is recently playing the role R with respect to the entities that share the long-term key K, and is possessing the fresh key K_s (if K_s is present)".

In order to enforce this interpretation of authentication blocks, we propose a number of principles to be followed by the trusted parties of an authentication protocol. These principles provide "safe ways" for generating new authentication blocks from existing ones. For example, an entity A can assign her identifier the Claimant role in order to authenticate herself with another entity B; also a trusted server S can assign to entity A the claimant role, in a message directed to B, provided that it has enough information to guarantee to B that A is indeed the claimant, i.e., provided that S has another authentication block proving that A is willing to authenticate with B.

We then show that authentication blocks enable one to reason on authentication. Specifically, we show that authentication guarantees can be expressed in terms of the decryption of particular authentication blocks. For example, if an entity B decrypts a block where A is playing the claimant role and B checks the nonce N present in the block, then he is guaranteed that A is indeed the identity of the claimant and that A was recently running the authentication protocol with him, i.e., A is authenticated to B.

We develop a formal model in which our principles are formalized as rules that regulate the formation of new authentication blocks from existing ones. Based on this model, we prove a safety theorem which implies that protocols designed according to these principles provide the intended authentication guarantees against any security attack to the protocol logic.

The paper is organized as follows. In Section 2, we give a precise account of the intended interpretation of authentication blocks, and introduce the design principles that convey that interpretation. In Section 3, we develop a formal model for protocol authentication and cast our principles within that model. Based on this formalization, we prove our safety theorem. In Section 4, we show that our principles are effective for reasoning on authentication protocols. In particular, we show that they may be exploited to re-arrange existing protocols, simplifying their structure. We conclude in Section 5 with a discussion of related work and final remarks. The proof of the safety theorem is detailed in a separate Appendix.

2 Authentication principles

We start by introducing our principles informally, giving the underlying ideas and motivating them through some examples. In Section 3, we formalize them as inference rules, proving their correctness with respect to a formal definition of entity authentication.

As we mentioned, we will only consider attacks which are based on flaws in the protocol logic and do not exploit any weakness of the underlying cryptographic primitives. We thus make the so called *perfect cryptography* assumption, i.e. we abstract away from security issues regarding cryptographic primitives and specific cryptographic algorithms, and we assume every cryptographic operation to be a secure building-block. Specifically, we assume that:

- encryption and decryption are possible only by knowing the correct keys;
- encrypted messages contain enough redundancy to make it possible to tell a meaningful message from a nonsensical one (this allows to detect the success (or failure) of a *decrypt* operation);
- long-term keys are safely distributed to participants and can never be learnt by any intruder (i.e., are stored locally in a secure place and are never sent out as protocol messages)

- every encrypted message has a form that makes it distinguishable from every other message in the protocol; this may be simply implemented by tagging encrypted messages with a message identifier, and is a simple way to avoid type-flaw attacks, i.e., attacks which are based on possible confusion among protocol messages [13].

Authentication protocols can be partitioned into two classes depending on whether or not they are based on a *Trusted Third Party* (TTP), i.e., a trusted entity that acts as intermediate party for the authentication task. The presence of a TTP greatly simplifies the key-management of long-term keys: it allows to have only one long-term key shared between each principal and the TTP, instead of having one long-term key for every possible pairs of principals. As we shall see, our principles allow us to deal with both kinds of authentication in a uniform way.

The principles are stated in terms of which roles encrypted components play in the authentication task. In particular we focus on three fundamental roles, and on the principles aiming at giving a correct "semantics" to such roles, by specifying (*i*) who is allowed to generate encrypted messages with specific roles, and (*ii*) under which conditions such a generation is possible. This controlled use of roles makes their interpretation unique, thus allowing use to derive authentication guarantees from the observation of encrypted messages. As an example, the reception of an encrypted message with A in the claimant role (and with a fresh nonce N) should imply that either A has generated such a fresh message or a trusted server S has "forwarded" an authentication request from A to B. The specific roles that entity identifiers can play are:

1. *Claimant*: an identifier representing a principal willing to authenticate itself to another principal;
2. *Intended verifier*: an identifier representing a principal that a claimant wants to authenticate with;
3. *Key owner*: an identifier representing a principal that owns a session key.

Roles are made explicit through *authentication blocks*, i.e., abstractions of an encrypted message which only contains the message components which are useful for authentication, plus their role in the authentication task. More in detail, an authentication block assigns a role to an entity identifier I, a nonce N and, possibly, a session key K_s. The nonce has the aim of proving that I was *recently* playing the specific role. The session key, if present, indicate that I (also) possesses the fresh key K_s.

Definition 1. *An authentication block is a tuple of the form* $\langle \mathsf{R}, I, N, K_s \rangle_{K_{I_1 I_2}}$ *where* $K_{I_1 I_2}$ $(I_1 \neq I_2)$ *is a long-term key shared between* I_1 *and* I_2*,* R *is one of the the labels* Claim *("Claimant"),* Owner *("Key Owner"),* Verif *("Intended Verifier"), I represents an entity playing such a role, N is a nonce, and* K_s*, if present, is a session key owned by I. The session key* K_s *is mandatory only for the role* Owner. *When it is missing the block is simply denoted by* $\langle \mathsf{R}, I, N \rangle_K$

Clearly, both I_1 and I_2 may decrypt[1] the block $\langle \mathsf{R}, I, N, K_s \rangle_{K_{I_1 I_2}}$. The intended semantics of authentication blocks is the following: if entity I_1 decrypts an authentication block and checks the validity of nonce N, then it is guaranteed that

> "Entity I is recently playing the role R with respect to I_1 and I_2, and is possessing the fresh key K_s (if K_s is present)".

where, in turn, "I is recently playing the role R with respect to I_1 and I_2" means:

1. if $\mathsf{R} = \mathsf{Claim}$, I_1 is not a TTP and $I_1 \neq I$ then I_1 is guaranteed that I is recently willing to authenticate with I_1;

[1] Notice that we are assuming $I_1 \neq I_2$, thus disregarding all issues related to self-authentication.

2. if $\mathsf{R} = \mathsf{Verif}$ I_1 is a TTP and $I_1, I_2 \neq I$ then I_1 is guaranteed that I_2 is recently willing to authenticate with I;
3. if $\mathsf{R} = \mathsf{Owner}$ then I_1 is guaranteed that I knows the session key K_s;

Notice that we give semantics to authentication blocks only in some specific situations. In particular, since the Claim role is used to authenticate I to I_1 (possibly through a TTP), we require I_1 to be a party other than a TTP and that $I_1 \neq I$, i.e., that I_1 does not accept authentication requests from itself (which is a typical source of "reflection attacks"). As the Verif role is used by I_2 to communicate to a TTP I_1 that it (I_2) intends to authenticate with I, we ask that I_1 is a TTP and that I is different from the other two identities. The fact that all the other cases have no semantics simply means that they do not provide any guarantee to the receiver.

Example 1. We show, through a simple example, that authentication blocks allow to reason about messages by making their interpretation unambiguous. Consider a message $\{M,A,N\}_{K_{BS}}$ encrypted with a long-term key shared between principal B and TTP S, and containing a message M, the entity identifier A and a nonce N. One can deduce that this message is probably used to authenticate something, as a nonce is present, but nothing more can be said from the message itself. If instead, we represent it as an authentication block, we recover a precise, and unambiguous semantics. For example we can represent $\{M,A,N\}_{K_{BS}}$ as the block $\langle \mathsf{Claim}, A, N\rangle_{K_{BS}}$, meaning that this message can be used to authenticate entity A to B, once nonce N has been checked by B. Alternatively, $\{M,A,N\}_{K_{BS}}$ could be mapped to $\langle \mathsf{Verif}, A, N\rangle_{K_{BS}}$ to represents a request from B to authenticate with A through the TTP S. Note that we cannot map the message to a block of the form $\langle \mathsf{Owner}, A, N, K_s\rangle_{K_{BS}}$ as no session key K_s is present in the message and K_s is mandatory for role Owner. In other words, to see a message as an authentication block, we need that every element required by the block is present in the original message. Note also that M is "discarded" in authentication blocks. This reflects the fact that it is irrelevant for authentication.

2.1 Protocols without TTP: Direct Authentication

If no TTP is present, the only way for Alice to authenticate with Bob is to use a long-term key shared between Alice and Bob. For any pair of principals A and B, we note K_{AB} the long-term key shared between them.

Principle 1 (Direct authentication) *Principal A may declare her intention to authenticate with principal B by generating an authentication block encrypted with K_{AB}, in which A plays the* Claim *role, i.e., a block of the form* $\langle \mathsf{Claim}, A, N\rangle_{K_{AB}}$.

Note that this principle is consistent with the intended semantics of the Claim role. When B decrypts the block $\langle \mathsf{Claim}, A, N\rangle_{K_{AB}}$, and verifies the nonce N, it is guaranteed that A is recently willing to authenticate with himself; moreover, if A receives such a message, she will discard it as the block is declaring herself as claimant.

Example 2. Consider the following simple protocol for unilateral authentication without TTP, inspired to the ISO Simmetric Key Two-Pass Unilateral Authentication Protocol [5, 15]:

$$\begin{array}{l} 1)\ B \rightarrow A : N_B \\ 2)\ A \rightarrow B : \{M,A,N_B\}_{K_{AB}} \end{array}$$

B, the verifier, sends a freshly generated nonce to A as a challenge. A, the claimant, completes the authentication session by sending a message M, A and N_B encrypted using K_{AB}. This proves to B the identity of the claimant A, as only A and B know K_{AB}.

As identifier A represents the *claimant* of the protocol run, we make this information explicit by mapping message 2 into the authentication block $\langle \mathsf{Claim}, A, N_B \rangle_{K_{AB}}$ (note that M is not in the block as it is unessential for authentication). The protocol is a "good one" as it follows principle 1.

Example 3. We now show that failing to comply with the previous principle may lead to protocol flaws. In particular, we consider the cases in which either A or N_B are missing in Message 2. Interestingly, in both both cases there is no way to represent the message as a valid authentication block as one of the fundamental components is missing. And in both cases the protocol is insecure. Suppose first that A is missing: Message 2 becomes $\{M, N_B\}_{K_{AB}}$. Observing message $\{M, N\}_{K_{AB}}$, there is no way to tell who between A and B is the claimant, so it is impossible to distinguish between messages that belong to a protocol run and messages that belong to another concurrent protocol run with reversed roles. This fact can be used effectively by an adversary as follows:

$$
\begin{array}{l}
1.a)\ B \rightarrow E(A) : N_B \\
1.b)\ E(A) \rightarrow B : N_B \\
2.a)\ B \rightarrow E(A) : \{M, N_B\}_{K_{AB}} \\
2.b)\ E(A) \rightarrow B : \{M, N_B\}_{K_{AB}}
\end{array}
$$

With such a "reflection attack" an enemy can trick B into believing that A has been actively involved in the protocol run "a", while in fact it has not.

Suppose now that the nonce is missing, i.e., the protocol becomes $\{A, M\}_{K_{AB}}$ (the first message is now useless). There is no way for B to tell if the message belongs to the current run or to a previous one, so an adversary can convince B that it is talking to A without A actively participating:

$$
\begin{array}{lll}
\text{old)} & A \rightarrow B & : \{A, M\}_{K_{AB}} \\
 & \ldots & \\
\text{new)} & E(A) \rightarrow B & : \{A, M\}_{K_{AB}}
\end{array}
$$

2.2 Protocols with TTP: indirect authentication

When a TTP is present, Alice may authenticate to Bob using the TTP as a mediator. In this setting, we assume that Alice and Bob share a long-term key with the TTP named S. Instead, there is no way for them to directly communicate through a common long-term key. The role Verif aims at dealing with this kind of indirect authentication:

Principle 2 (Authentication through TTP) *Principal A may declare to a TTP S her intention to authenticate with principal B by generating an authentication block encrypted with K_{AS}, in which B plays the* Verif *role, i.e., a block of the form $\langle \mathsf{Verif}, B, N \rangle_{K_{AS}}$.*

Note that this principle is consistent with the intended semantics of Verif role: it allows A to communicate to a TTP her intention of authenticating with B.

This principle is useful when combined with other principles that allow the TTP to forward authentication to the intended verifier B.

Principle 3 (Verified authentication forwarding) *TTP S may declare to a principal B that a principal A is recently willing to authenticate with B, provided that S has evidence of that fact, i.e, has* *decrypted a block* $\langle \mathsf{Verif}, B, N\rangle_{K_{AS}}$ *and checked the validity of* N*. In such a case,* S *generates an authentication block encrypted with* K_{BS}*, in which* A *plays the* Claim *role, i.e., a block of the form* $\langle \mathsf{Claim}, A, N'\rangle_{K_{BS}}$.

The principle above states that a TTP can forward authentication from A to B if the TTP is guaranteed that A is recently willing to authenticate with B. This is provided by the (intended) semantics of the authentication block $\langle \mathsf{Verif}, B, N\rangle_{K_{AS}}$, and this principle is thus consistent with the intended semantics of $\langle \mathsf{Claim}, A, N'\rangle_{K_{BS}}$.

Example 4 (nonce-based Wide Mouth Frog Protocol[2]*).* The Wide Mouth Frog protocol [7] achieves unilateral authentication using a TTP. The original version is based on timestamps, here we consider the nonce-based version presented in [11].

$$
\begin{array}{l}
1)\ A \rightarrow S : A \\
2)\ S \rightarrow A : N_S \\
3)\ A \rightarrow S : A, \{B, K_{AB}, N_S\}_{K_{AS}} \\
4)\ S \rightarrow B : () \\
5)\ B \rightarrow S : N_B \\
6)\ S \rightarrow B : \{A, K_{AB}, N_B\}_{K_{BS}}
\end{array}
$$

This protocol can be seen as a direct application of principles 2 and 3. As a matter of fact, the two encrypted messages 3 and 6, can be abstracted as the two authentication blocks $\langle \mathsf{Verif}, B, N_S\rangle_{K_{AS}}$ and $\langle \mathsf{Claim}, A, N_B\rangle_{K_{BS}}$, respectively. This last block provides an authentication guarantee to B. Note that the two encrypted messages have the same form. Assigning them to two different authentication blocks make their interpretation very different and allows to easily reason about which guarantees the two messages provide.

Next principle captures a variant of authentication forwarding that allows the TTP to forward authentication even if it has not verified the claimant's nonce. The idea is that the nonce sent by the claimant is forwarded to the verifier who is in charge of checking its validity.

Principle 4 (non-verified authentication forwarding) *TTP* S *may declare to a principal* B *that a principal* A *is willing to authenticate with* B *with a nonce* N*, provided that* S *has evidence of that fact, i.e, has decrypted a block* $\langle \mathsf{Verif}, B, N\rangle_{K_{AS}}$*. In such a case,* S *generates an authentication block encrypted with* K_{BS}*, in which* A *plays the* Claim *role and with the same nonce* N *of the previous block, i.e., a block of the form* $\langle \mathsf{Claim}, A, N\rangle_{K_{BS}}$.

The difference with respect to principle 3 is that N is not checked by the TTP but is forwarded to B, in order to let him check its validity. This principle is again consistent with the intended semantics of authentication blocks: the TTP is assured that A is willing to authenticate with B even if it does not know anything about the freshness of this intention; for this reason, the TTP forwards the nonce N to B so that B may check it and verify that the willingness of A to authenticate with him is also recent.

A possible scenario is the following: the nonce is originated by the verifier, sent to the claimant who inserts it into the authentication request to the TTP; finally the nonce is forwarded back to the verifier for the final check. This is what the "Woo and Lam Shared-Key Protocol" does [26].

[2] This protocol suffers of a *type-flaw* attack: assuming that ciphertexts in message 3 and message 6 can be confused (this is the base of a type-flaw) and both the principals A and B can play both roles, an adversary can succeed impersonating one of the principals (see [11, 4, 6] for more detail). Recall that we are assuming that confusion among messages is not possible, i.e., that encrypted protocol messages are implemented in a way that allow to distinguish among them, thus preventing the attack mentioned above.

Example 5 (Flaws in the original Woo and Lam protocol). Here we report the original (flawed) version of the protocol [26].

$$
\begin{array}{l}
1)\ A \rightarrow B : A \\
2)\ B \rightarrow A : N_B \\
3)\ A \rightarrow B : \{N_B\}_{K_{AS}} \\
4)\ B \rightarrow S : \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
5)\ S \rightarrow B : \{N_B\}_{K_{BS}}
\end{array}
$$

Message 3 is encrypted with the key shared with S and is thus directed to S (even if it is sent to B who is in charge of forwarding it to S inside message 4). It represents the declaration of the intention of A to authenticate with B. However, note that A does not include in the ciphertext of message 3 the label of B. Thus, it is not possible to apply principle 2. The same happens in the last message, where S doesn't include any identity label. This make principle 4 inapplicable.

This failure to comply with the principles suggests following attack:

$$
\begin{array}{l}
\alpha.1)\ A \rightarrow E : A \\
\beta.1)\ E \rightarrow B : A \\
\beta.2)\ B \rightarrow E : N_B \\
\alpha.2)\ E \rightarrow A : N_B \\
\alpha.3)\ A \rightarrow E : \{N_B\}_{K_{AS}} \\
\beta.3)\ E \rightarrow B : \{N_B\}_{K_{AS}} \\
\beta.4)\ B \rightarrow S : \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
\beta.5)\ S \rightarrow B : \{N_B\}_{K_{BS}}
\end{array}
$$

At the end of the attack sequence B, accepts the authentication request of A even if A began a session to authenticate itself with E, i.e., there is no correspondence of intentions. This is considered a valid attack to entity authentication by many authors (see, e.g., [27, 16]). Interestingly, the attack exploits the absence of B as intended verifier in message 3. To see that, note that $\beta.3$, E forwards to B the message sent by A who is willing to authenticate with E. If the intended verifier (E in this case) were made explicit in the message, such a forwarding would be detected by the server before sending out the last message $\beta.5$. In fact, the correct version of this protocol [11] uses the identifiers precisely as required by our principles.

2.3 Protocols with TTP based on session keys

Another way for Alice to authenticate with Bob through a TTP, is to establish a fresh session key with Bob and then use it to authenticate her. The following principle states how key distribution should happen:

Principle 5 (Session key distribution) *TTP S may distribute to principals A and B a freshly generated key K_s by generating two authentication blocks encrypted with K_{AS} and K_{BS} and with B and A playing the* Owner *role, respectively. In particular, the two blocks have the form $\langle \mathsf{Owner}, B, N, K_s \rangle_{K_{AS}}$ and $\langle \mathsf{Owner}, A, N', K_s \rangle_{K_{BS}}$, respectively. No other messages containing K_s may be sent by S.*

The principle above shows how the Owner role is used: the TTP communicates to A (B) that B (A) owns the key K_s. Note that, in order to guarantee the secrecy of the session key, the principle requires that the TTP will never send other messages containing K_s.

Remark 1 (Role subsumption). It could be the case that a TTP generates, through principles 3 and 4, a block where A plays the Claim role. At the same time the TTP could need to distribute to B a session key owned by A, by applying principle 5 above. As a result the TTP generates the two blocks $\langle \mathsf{Claim}, A, N \rangle_{K_{BS}}$ and $\langle \mathsf{Owner}, A, N', K_s \rangle_{K_{BS}}$. Since the information conveyed by the two blocks does not overlap, we allow the role Claim to subsume role Owner, provided that K_s is specified in the corresponding block. In other words, instead of generating the two blocks above, the TTP may just generate $\langle \mathsf{Claim}, A, N, K_s \rangle_{K_{BS}}$.

The effect is to have in just one encrypted message the information conveyed by the two separated ones: "A recently plays the Claim role and possesses the fresh key K_s". Note that this is still consistent with the (intended) semantics of authentication blocks. A known protocol, using this idea, is, for instance, the Yahalom Protocol [7]. In section 4, we show an optimized variant of the Needham-Schroeder Shared-Key Protocol that is similar to (a correct version of) the Yahalom Protocol.

The next principle is the basic step to provide entity authentication through session keys

Principle 6 (Authentication through session-keys) *Principal A may declare her intention to authenticate with principal B by generating a ciphertext $\{M\}_{K_s}$, provided that A has evidence that K_s is a fresh key owned by B, i.e., has decrypted a block $\langle \mathsf{Owner}, B, N, K_s \rangle_{K_{AS}}$ and checked the validity of N.*

Applying this principle, a principal B may conclude that A is recently willing to authenticate with him, if he receives a message $\{M\}_{K_s}$ and knows that (i) K_s is a fresh key owned by A and (ii) he did not generate $\{M\}_{K_s}$. An example of a protocol based on this kind of authentication is the Amended Needham Schroeder Shared-Key Protocol [19].

2.4 Authentication Guarantees

In view of the principles we have outlined, there are two situations that can drive an entity to accept an authentication request, which we formalize next. In the following definition, we tacitly assume that B is not a TTP.

Definition 2 (Authentication 1). *An entity B accepts the authentication request of A whenever it receives and verifies (the nonce of) an authentication block $\langle \mathsf{Claim}, A, N \rangle_{K_{IB}}$, or $\langle \mathsf{Claim}, A, N, K \rangle_{K_{IB}}$, with I any entity other than B.*

Note that if I is A, then the authentication request is *directly* made by A. Otherwise, I must be a TTP, say S, and the request is indirectly made S on behalf of A. The definition clarifies the importance of the role Claim in describing the identity labels used to communicate a trusted authentication request.

There is, however, another way that an entity can to accept an authentication request:

Definition 3 (Authentication 2). *An entity B accepts the authentication request of A whenever it has received and verified (the nonce of) an authentication block in the form $\langle \mathsf{Owner}, A, N, K_s \rangle_{K_{SB}}$, with S being a TTP, and receives a message $\{M\}_{K_s}$ that he did not generate.*

3 Authentication Principles, Formally

We formalize our principles as a set of deduction rules, which specify (i) the format of the ciphertexts that a trusted party should generate to complete an authentication session, and (ii) the conditions under which such ciphertexts can safely be generated. The ciphertexts can be in two forms: either authentication blocks encrypted with long-term keys (which we assume to be robust) or any other message encrypted with session keys which are possessed by the trusted parties, and are only circulated among them within authentication blocks.

Remarkably, the rules predicate the actions by the trusted parties only on the structure of the ciphertexts that can be *observed* on the network. Thus, no knowledge is assumed on where the traffic originates: it may either be generated by the trusted parties of the protocol, or else be the result of the attacker generating its own traffic, or duplicating existing messages. It is important to note that observing network traffic effectively corresponds to assuming that all the ciphertexts reach *all* parties, included their intended recipients. This may at first be understood as a limitation on the power of the intruder, because it prevents attacks to be mounted based on the interception (i.e., subtraction from the net) of ciphertexts. This is not the case, however, as intercepting a message may only break liveness properties, such as *fairness* in contract-signing, where two parties wants to be guaranteed that none of them will get the contract signed before the other one (see, e.g., [22, 28]). Authentication, instead, is (formalized as) a *safety* property, which is immune to attacks based on message absorption.

On the other hand, observing network traffic is appealing, as it simplifies our formalization, because it allows us to represent any protocol run solely in terms of the trace of the the outgoing traffic, disregarding message reception.

3.1 Environments and Traces

The deduction rules derive judgments of the form $\Gamma \vdash \sigma$, stating that σ is a *trace*, relative to the *environment* Γ. The structure of environments and traces is summarized in Table 1.

Table 1 Environments and Traces

Events		*Environments*	
$e ::= new(I,V)$	New nonce/key	$\Gamma ::= \emptyset$	empty
		$\mid\ \Gamma, e$	new nonce/key
Markers		*Traces*	
$\mu ::= e$	event	$\sigma ::= \varepsilon$	empty
$\mid\ run_{\mathsf{R}}(I_1,I_2)$	init	$\mid\ \alpha :: \sigma$	*actions*
$\mid\ commit(I_1,I_2)$	completion		
$\mid\ check(I,N)$	nonce check		
Actions			
$\alpha ::= \mu$	marker		
$\mid\ I \triangleright \{M\}_K$	encrypted message		
$\mid\ I \triangleright \langle \mathsf{R}, I, N, K_S \rangle_K$	authentication block		

We presuppose two sets of party identity labels: **S** for Trusted Third Parties (or *Servers*), ranged over by S, and **P** for principals, ranged over by A, B, and two sets **K** of keys and **N** of nonces, ranged over by K and N respectively. All such sets are assumed to be pairwise disjoint. We let I range over $\mathbf{S} \cup \mathbf{P}$, and V over $\mathbf{K} \cup \mathbf{N}$. Finally, R ranges over the roles $\mathsf{Claim}, \mathsf{Verif}$ and Owner.

A trace σ is a sequence of actions and collects the encrypted messages generated during (possibly parallel) protocol runs as well as additional *marker* actions that we use to prove our safety result. In $\Gamma \vdash \sigma$, Γ keeps track of the *events* of protocol runs, i.e. the generation of the nonces and of the session keys used in the runs.

The intuitive reading of the actions is as follows. Event $new(I,V)$ traces the generation of the nonce or key V by the entity I. Marker $check(I,N)$ indicates the checking of the nonce N by the entity I. Marker $run_{\mathsf{R}}(I_1,I_2)$ indicates the intention of (equivalently, the start of a protocol run by) I_1 to authenticate itself with I_2, by exploiting the role R. Note that there might be many $run_{\mathsf{R}}(I_1,I_2)$ markers corresponding to many (either parallel or sequential) protocol sessions. Finally, $commit(I_1,I_2)$ marks the completion of the authentication session. The remaining actions denote the generation of encrypted messages and of authentication blocks by an entity I: specifically, $I \triangleright \{M\}_K$ and $I \triangleright \langle \mathsf{R}, I, N, K_S \rangle_K$ represent the generation by entity I of $\{M\}_K$ and $\langle \mathsf{R}, I, N, K_S \rangle_K$, respectively.

We use a number of notation conventions and shorthands. We write $run(B,A)$ instead of $run_{\mathsf{R}}(B,A)$ when the role R is immaterial. We write authentication blocks as $\langle \mathsf{R}, I, N \rangle_K$ when $\mathsf{R} \in \{\mathsf{Claim}, \mathsf{Verif}\}$, as shorthands for $\langle \mathsf{R}, I, N, * \rangle_K$, where $*$ represents an empty key field.

A further definition is useful in presenting the deduction rules. We introduce an ordering relation over events. The ordering is defined in terms of two corresponding orderings for keys and roles, noted $\leq$, and defined as the least reflexive relations such that $K \leq *$, and $\mathsf{Claim} \leq \mathsf{Owner}$, $\mathsf{Verif} \leq \mathsf{Owner}$, respectively. The ordering over actions is also noted $\leq$, and defined as the least transitive and reflexive relation such that $I \triangleright \langle \mathsf{R}_1, I, N, K_1 \rangle_K \leq I \triangleright \langle R_2, I, N, K_2 \rangle_K$ iff $\mathsf{R}_1 \leq \mathsf{R}_2$ and $K_1 \leq K_2$.

Definition 4 (block $\in$ trace). *Given an action α and a trace σ, we write $\alpha \in \sigma$ iff there exists an action $\alpha \leq \alpha'$ such that $\sigma = \sigma' :: \alpha' :: \sigma''$.*

Intuitively, for an action α to belong to a trace σ, it is sufficient that a "greater" (or equal) action α' belongs to σ.

3.2 Deduction Rules

The deduction rules are reported in Table 2 and have the following uniform structure:

$$\frac{\Gamma \vdash \sigma \quad (\textit{side conditions})}{\Gamma' \vdash \sigma'}$$

where σ' extends σ. They define the conditions under which the judgment in the conclusion may be derived from the judgment in the premise by including further actions in the trace. As anticipated, the side conditions only test the presence of certain ciphertexts, and marker actions, in the current trace σ. We do not assume any knowledge on the originators of the ciphertexts, which may therefore be generated by trusted parties or the intruder[3]. As for the marker actions, the side conditions test the presence of events of the form $new(I,V)$, and of markers $run_{\mathsf{R}}(I_1,I_2)$. The former represents the creation of nonces and session keys, whose presence can be tested locally by the originators. The latter, instead, mark the authentication intentions of the trusted parties of the protocol. These markers are not part of the protocol: we include them in our traces to give a formal proof of safety. In doing so, we assume that the protocol runs always start in the same way (hence applying the same principle), i.e., if two protocol runs are indicated by two markers $run_{\mathsf{R}}(I_1,I_2)$ and $run'_{\mathsf{R}}(I'_1,I'_2)$ in the same trace σ, then it must be $R = R'$. This is reasonable as two sessions of the same protocol always start with the same message, even if the entities may be different or could exchange their roles.

The environment Γ collects the *events* of the protocols, i.e. the creation of session keys and nonces, and it is used by the deduction rules to ensure the freshness of nonces and session keys. The intuition is as follows: when a new nonce is generated, or a session key created, it is included in the environment (and in the trace). A nonce (key) can only be generated if it does not occur in the current trace (it is fresh). Nonces can be checked, (keys used) only if they are available in the environment, and once

[3] With one exception (cf. rule AUTHENTICATION 2).

a nonce is checked (a key used) the corresponding event is discarded from the environment (but not from the trace) and hence it is never available again for checking (usage).

We proceed with the description of the deduction rules.

Rule (EMPTY) is the only axiom. It is used to guarantee that every derivation (hence execution) starts with an empty set of already generated nonces and session keys: nonces and sessions keys must be generated during the protocol runs. (NEW) formalizes nonce/key creation. The notation *V fresh in* σ indicates that there is no I such that $new(I,N) \in \sigma$: in other words, V must be fresh with respect to all the previous protocol runs. As we mentioned above, (and as we shall prove) the deduction rules guarantee that fresh keys are used (and nonces are checked) at most once.

(PRINCIPLE 1) formalizes the principle of direct authentication. A sends an authentication block with A as Claimant to B. This is the start of an authentication session, which is marked by the inclusion of $run_{\mathsf{Claim}}(A,B)$ in the current trace. (PRINCIPLE 2) is similar. Only, it marks the start of the authentication session with $run_{\mathsf{Verif}}(A,B)$ and represents the start of an authentication session through a TTP. In both cases, A may be running other authentication sessions, provided that they are all marked with either run_{Claim} or run_{Verif}. (PRINCIPLE 3) and (PRINCIPLE 4) formalize the two principles of authentication forwarding: the former formalizes verified forwarding, by checking the nonce, the latter unverified forwarding. In particular, to guarantee that N is never used again in the execution, (PRINCIPLE 3) discards the event $new(S,N)$ from Γ. In (PRINCIPLE 5) the server S distributes a fresh session key to the two parties A and B. The event $new(S,K)$ is discarded from Γ to ensure that K is never used again in the current session. (PRINCIPLE 6) allows A to send a message encrypted with a session key K. The condition $I \triangleright \langle \mathsf{Owner},B,N,K\rangle_{K_{AS}} \in \sigma$ binds K to the principal which shares K.

Two rules formalize the conditions for accepting authentication requests. They reflect the two way to accept authentication presented earlier in Definitions 2 and 3. A further remark is in order for the side-condition $I \neq B$ in (AUTHENTICATION 2). Here we are assuming that B, which is *committing* (hence completing the authentication session) has the ability to tell that he is not the originator of the ciphertext $\{M\}_K$. This is can easily be accomplished as done in the Amended Needham Schroeder Shared-Key Protocol [19].

3.3 Safety

The purpose of the deduction rules is to generate *safe* traces, that is traces that provide the intended authentication guarantees for a protocol session. We formulate these guarantees as a safety property for traces following the idea of *correspondence* between actions of the participants [27, 16]. Intuitively, a trace is safe if every $commit(A,B)$ in the trace is preceded by a corresponding $run(B,A)$ action. Formally:

Definition 5 (Safety). *A trace* σ *is safe if and only if whenever* $\sigma = \sigma_1 :: commit(B,A) :: \sigma_2$, *one has* $\sigma_1 = \sigma_1' :: run(A,B) :: \sigma_1''$, *and* $\sigma_1' :: \sigma_1'' :: \sigma_2$ *is safe.*

We conclude with the main result of this section, stating that protocol protocols designed according to our principles do indeed achieve authentication. Formally, we have:

Theorem 1 (Safety). *If* $\Gamma \vdash \sigma$ *then* σ *is safe.*

Proof. Follows as a corollary of Theorem 2 (in Section 4).

Table 2 Derivation Rules

Principles

$$\text{EMPTY}\quad \frac{}{\emptyset \vdash \varepsilon} \qquad \text{NEW}\quad \frac{\Gamma \vdash \sigma \qquad V \textit{ fresh in } \sigma \qquad V \in \mathbf{K} \cup \mathbf{N}}{new(I,V), \Gamma \vdash \sigma :: new(I,V)}$$

$$\text{PRINCIPLE 1}\quad \frac{\Gamma \vdash \sigma \qquad run_R(A,B) \in \sigma \Rightarrow R = \mathsf{Claim}}{\Gamma \vdash \sigma :: run_{\mathsf{Claim}}(A,B) :: A \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{AB}}} \qquad \text{PRINCIPLE 2}\quad \frac{\Gamma \vdash \sigma \qquad run_R(A,B) \in \sigma \Rightarrow R = \mathsf{Verif}}{\Gamma \vdash \sigma :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}}}$$

$$\text{PRINCIPLE 3}\quad \frac{new(S,N), \Gamma \vdash \sigma \qquad I \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} \in \sigma}{\Gamma \vdash \sigma :: check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N_A \rangle_{K_{BS}}} \qquad \text{PRINCIPLE 4}\quad \frac{\Gamma \vdash \sigma \qquad I \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} \in \sigma}{\Gamma \vdash \sigma :: S \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{BS}}}$$

$$\text{PRINCIPLE 5}\quad \frac{\Gamma, new(S,K) \vdash \sigma}{\Gamma \vdash \sigma :: S \triangleright \langle \mathsf{Owner}, A, N_A, K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N_B, K \rangle_{K_{AS}}}$$

$$\text{PRINCIPLE 6}\quad \frac{\Gamma \vdash \sigma \qquad I \triangleright \langle \mathsf{Owner}, B, N, K \rangle_{K_{AS}} \in \sigma \qquad run_R(A,B) \in \sigma \Rightarrow R = \mathsf{Owner}}{\Gamma \vdash \sigma :: run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K}$$

Authentication Guarantees

$$\text{AUTHENTICATION 1}\quad \frac{new(B,N), \Gamma \vdash \sigma \qquad J \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{IB}} \in \sigma \qquad I \in \{S, A\}}{\Gamma \vdash \sigma :: check(B,N) :: commit(B,A)}$$

$$\text{AUTHENTICATION 2}\quad \frac{new(B,N), \Gamma \vdash \sigma \qquad I \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{SB}} \in \sigma \qquad J \triangleright \{M\}_K \in \sigma \qquad J \neq B}{\Gamma \vdash \sigma :: check(B,N) :: commit(B,A)}$$

4 Principles at work: an example

The Needham-Schroeder protocol [21, 19, 9] achieves mutual authentication with session key exchange based on a TTP named S:

$$\begin{array}{l}
1)\ A \rightarrow B : A \\
2)\ B \rightarrow A : \{A, N_B^0\}_{K_{BS}} \\
3)\ A \rightarrow S : A, B, N_A, \{A, N_B^0\}_{K_{BS}} \\
4)\ S \rightarrow A : \{B, K_s, N_A, \{A, K_s, N_B^0\}_{K_{BS}}\}_{K_{AS}} \\
5)\ A \rightarrow B : \{A, K_s, N_B^0\}_{K_{BS}} \\
6)\ B \rightarrow A : \{N_B\}_{K_s} \\
7)\ A \rightarrow B : \{N_B - 1\}_{K_s}
\end{array}$$

The logic of the protocol can be expressed in terms of our principles as follows. Consider Message 4 first. The authentication blocks corresponding to the two encryptions are: $\langle \mathsf{Owner}, B, N_A, K_s \rangle_{K_{AS}}$ and $\langle \mathsf{Owner}, A, N_B^0, K_s \rangle_{K_{BS}}$ Now, B and A after checking the respective nonces may, by virtue of Principle

6, generate messages 6 and 7, respectively. Mutual authentication is provided by two applications of the authentication guarantee of Definition 3 [4].

As a further remark, note that the identifier A in the ciphertext $\{A,N_B^0\}_{K_{BS}}$ of Messages 2 and 3, represents the *intended verifier* of the authentication run between B and A. The authentication block corresponding to the message is thus $\langle \mathsf{Verif}, A, N_B^0\rangle_{K_{BS}}$. Notice that this block is not necessary to prove authentication. This suggests that such an encryption could be safely eliminated to obtain a protocol which is very similar to Carlsen's Secret Protocol [8, 9].

$$
\begin{array}{l}
1)\ A \to B : A\\
2)\ B \to A : A, N_B^0\\
3)\ A \to S : A, B, N_A, N_B^0\\
4)\ S \to A : \{B, K_s, N_A, \{A, K_s, N_B^0\}_{K_{BS}}\}_{K_{AS}}\\
5)\ A \to B : \{A, K_s, N_B^0\}_{K_{BS}}\\
6)\ B \to A : \{N_B\}_{K_s}\\
7)\ A \to B : \{N_B - 1\}_{K_s}
\end{array}
$$

Now we can show the trace observed at the end of the protocol in which were envolved two principals, A and B, and a trusted third party S.

$\emptyset \vdash new(B,N_B) :: new(A,N_A) :: S \triangleright \langle \mathsf{Owner}, B, N_A, K_s\rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N_B^0, K_s\rangle_{K_{BS}} :: run_{\mathsf{Owner}}(B,A) :: B \triangleright \{M\}_{K_s} :: check(A,N_A) :: commit(A,B) :: run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_{K_s} :: check(B,N_B) :: COMMIT(B,A)$

Note that we can't deal with nested authentication blocks. So the nested ciphertexts of the fourth message exchange are translated into two distinct authentication blocks. In this protocol the nested ciphertext of message 4 assures B of the presence of A, but it doesn't involve any guarantee on the authentication intention of A. In our opinion it's necessary considering that A could send such a ciphertext even if the label B, of message 4, doesn't correspond to its intended verifier. Note also that the presence of A is assured by the seventh message exchange. Such considerations lead us to ignore nested ciphertexts by coding them into distinct authentication blocks.

Below, we discuss two variants that result from an application of our principles.

First note that, as observed in Remark 1 (on page 9) it is safe to overlap the Claim and Owner roles if we need to say that "A recently plays the Claim role and possesses the fresh key K_s". This can be formalized by adding the two new deduction rules of Table 3.

Table 3 Derived Proof Rules

PRINCIPLE 5'

$$\frac{new(S,K), new(S,N), \Gamma \vdash \sigma \qquad I \triangleright \langle \mathsf{Verif}, B, N\rangle_{K_{AS}} \in \sigma}{\Gamma \vdash \sigma :: check(N) :: S \triangleright \langle \mathsf{Claim}, A, N_A, K\rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N_B, K\rangle_{K_{AS}}}$$

PRINCIPLE 5''

$$\frac{new(S,K), \Gamma \vdash \sigma \qquad I \triangleright \langle \mathsf{Verif}, B, N\rangle_{K_{AS}} \in \sigma}{\Gamma \vdash \sigma :: S \triangleright \langle \mathsf{Claim}, A, N, K\rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N_B, K\rangle_{K_{AS}}}$$

(PRINCIPLE 5') and (PRINCIPLE 5'') differ from the previous (PRINCIPLE 5) by considering that an entity label can be used with two roles, either claimant or key owner. S sends one of the two

[4] Recall that we are assuming that encrypted messages have enough redundancy to allow to detect the success of a decryption with a given key

authentication blocks with the idea of ensuring B that A is the claimant of the current session and that it shares with A a session key K. PRINCIPLE 5' uses, to assign to A the Claim role, the same hypotheses used in PRINCIPLE 3 . Similarly PRINCIPLE 5" uses the hypotheses of PRINCIPLE 4 .

The safety theorem still holds if we add these two new inference rules. In fact, we have:

Theorem 2 (Safety). *Extending the deduction system with (*PRINCIPLE *5') and (*PRINCIPLE *5") we still have that if* $\Gamma \vdash \sigma$ *then* σ *is safe.*

Proof. In Appendix A.

An application of (PRINCIPLE 5') enables the following optimized variant of the Needham-Schroeder Shared-Key Protocol[5]:

$$
\begin{aligned}
&1)\ A \to B : A, N_A \\
&2)\ B \to A : \{A, N_B^0, N_A\}_{K_{BS}} \\
&3)\ A \to S : A, B, \{A, N_B^0, N_A\}_{K_{BS}} \\
&4)\ S \to A : \{N_A, B, K_s, \{K_s, A, N_B^0\}_{K_{BS}}\}_{K_{AS}} \\
&5)\ A \to B : \{K_s, N_B^0, A\}_{K_{BS}},\ \{N_B\}_{K_s}
\end{aligned}
$$

The only difference with respect to the original version is the presence of the nonce N_A which is sent by A as cleartext to B who inserts it in the encrypted message $\{A, N_B^0, N_A\}_{K_{BS}}$.

The identifier A in the ciphertext $\{A, N_B^0, N_A\}_{K_{BS}}$ of Messages 2 and 3, represents the *intended verifier* of the authentication run between B and A. A possible authentication block corresponding to the message is $\langle \mathsf{Verif}, A, N_A \rangle_{K_{BS}}$. By applying (PRINCIPLE 5') we obtain the two blocks of authentications: $\langle \mathsf{Claim}, B, N_A, K_s \rangle_{K_{AS}}$ and $\langle \mathsf{Owner}, A, N_B^0, K_s \rangle_{K_{BS}}$. Differently from the previous protocol, here we obtain that B besides to owning the session key, is also playing the Claim role. This allows to obtain the authentication of B with respect to A by applying the authentication guarantee of definition 2. The mutual authentication is obtained as before through the last encrypted message $\{N_B\}_{K_s}$.

A further optimized variant for this protocol can be derived by applying (PRINCIPLE 5").

$$
\begin{aligned}
&1)\ A \to S : A, B \\
&2)\ S \to B : A, B, N_S \\
&3)\ B \to A : \{A, N_S\}_{K_{BS}}, N_B^0 \\
&4)\ A \to S : A, B, N_A, \{A, N_B^0, N_S\}_{K_{BS}} \\
&5)\ S \to A : \{N_A, B, K_s, \{K_s, A, N_B^0\}_{K_{BS}}\}_{K_{AS}} \\
&6)\ A \to B : \{K_s, N_B^0, A\}_{K_{BS}},\ \{N_B\}_{K_s}
\end{aligned}
$$

In this variant S sends a nonce to B to check the freshness of the authentication request of B. The identifier A in the ciphertext $\{A, N_B^0, N_S\}_{K_{BS}}$ represents again the *intended verifier* of the authentication run between B and A. A possible corresponding block of authentication is $\langle \mathsf{Verif}, A, N_S \rangle_{K_{BS}}$.

The authentication blocks corresponding to $\{N_A, B, K_s, \{K_s, A, N_B^0\}_{K_{BS}}\}_{K_{AS}}$ are build by applying (PRINCIPLE 5") and they are:

$$\langle \mathsf{Claim}, B, N_A, K_s \rangle \text{ and } \langle \mathsf{Owner}, A, N_B^0, K_s \rangle$$

Again B is playing the Claim role. As in the previous variant, after the sixth message exchange, the goals of mutual entity authentication are achieved.

[5] Note that the protocol looks like a correct version of the Yahalom Protocol

5 Conclusions

We have studied a natural, role-based interpretation of message components in cryptographic protocols, and we have shown that this interpretation contributes to the task achieving entity authentication.

The notion of authentication block we have proposed is interesting in two respects. First, it provides a flexible formal tool for reasoning on different protocols in uniform ways. Secondly, it enables the extraction of few fundamental mechanisms to be combined in the design of provably correct authentication protocols.

The formal model we have developed appears to be fairly general, but a number of extensions would be desirable to capture a wider class of protocols. We are currently studying the extensions needed to handle protocols that rely on nested encryption, and asymmetric keys.

Also, it would be interesting to develop methods for the static analysis of protocols based on our principles. As a prerequisite for that, is to re-cast our current development on existing operational models for cryptographic protocols like *Strand Spaces* [25, 23], *spi* calculus [1], CCS and CSP based models [10, 17]. Finally, relationships with approaches based on type systems [11, 2, 12, 2, 3], and logics [7, 24] deserve to be made.

References

1. M. Abadi and A. D. Gordon. "A Calculus for Cryptographic Protocols: The Spi Calculus". *Information and Computation*, 148(1):1–70, 1999.
2. Martín Abadi and Bruno Blanchet. Secrecy types for asymmetric communication. In *Foundations of Software Science and Computation Structures (FoSSaCS'01)*, volume 2030 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2001.
3. Martín Abadi and Bruno Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. In *29th Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages (POPL 2002)*, pages 33–44, Portland, Oregon, January 2002. ACM Press.
4. Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.
5. P.C. van Oorschot A.J.Menezes and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
6. Ross Anderson and Roger Needham. Programming satan's computer. In Jan van Leeuwen, editor, *Computer Science Today — Recent Trends and Developments*, volume 1000 of *ln-cs*, pages 426–440. 1995.
7. M. Burrows, M. Abadi, and R. Needham. "A Logic of Authentication". *Proceedings of the Royal Society of London*, 426(1871):233–271, 1989.
8. U. Carlsen. Generating formal cryptographic protocol specifications. In *IEEE Symposium on Research in Security and Privacy*, pages 137–146. IEEE Computer Society, 1994.
9. John A. Clark and Jeremy L. Jacob. A survey of authentication protocol literature. Technical Report 1.0, 1997.
10. R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Proceedings of ICALP'00*, pages 354–372. Springer LNCS 1853, July 2000.
11. A. Gordon and A. Jeffrey. Authenticity by typing for security protocols, 2001.
12. Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. In *15th IEEE Computer Security Foundations Workshop — CSFW'01*, pages 77–91, Cape Breton, Canada, 24–26 June 2002. IEEE Computer Society Press.
13. James Heather, Gavin Lowe, and Steve Schneider. How to prevent type flaw attacks on security protocols. In *13th IEEE Computer Security Foundations Workshop — CSFW'00*, pages 255–268, Cambridge, UK, 3–5 July 2000. IEEE Computer Society Press.
14. ISO/IEC. *Information Technology-Security Tecniques-Entity Authentication Mechanisms, Part 1:General Model*. 1991.
15. ISO/IEC. *Information Technology-Security Tecniques-Entity Authentication Mechanisms, Part 2:Entity Authentication using Simmetric Tecniques*. 1993.
16. G. Lowe. "A Hierarchy of Authentication Specification". In *Proceedings of the 10th Computer Security Foundation Workshop*. IEEE press, 1997.
17. G. Lowe. "CASPER: A Compiler for the Analysis of the security protocols". *Journal of Computer Security*, 6:53–84, 1998.

18. Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.
19. R M Needham and M D Schroeder. Authentication revisited. *ACM SIGOPS Operating Systems Review*, 21(1):7–7, 1987.
20. Lawrence C. Paulson. Relations between secrets: Two formal analyses of the yahalom protocol. *Journal of Computer Security*, 9(3):197–216, 2001.
21. M.D. Schroeder R.M. Needham. Using encryption for authentication in large networks of computers. *ACM Communication*, 21(12):993–999, 1978.
22. S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of CSFW'98*, pages 54–65. IEEE Press, 1998.
23. P. Syverson. Towards a strand semantics for authentication logics, 1999.
24. Paul Syverson and Iliano Cervesato. The logic of authentication protocols. *Lecture Notes in Computer Science*, 2171, 2001.
25. J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Proving security protocols correct, 1999.
26. Thomas Y. C. Woo and Simon S. Lam. Authentication for distributed systems, from computer, january, 1992. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1992*. 1992.
27. T.Y.C. Woo and S.S. Lam. "A Semantic Model for Authentication Protocols". In *Proceedings of 1993 IEEE Symposium on Security and Privacy*, pages 178–194, 1993.
28. J. Zhou and D. Gollman. A fair non-repudiation protocol. In *Proc. of Symposium in Research in Security and Privacy*, pages 55–61. IEEE Press, 1996.

A Proof of Safety

We first introduce some preliminary lemmas and notation. The prefix relation over traces is denoted $\sqsubseteq$, and we write $\sigma_1 \setminus \sigma_2$ for the trace obtained from σ_1 by dropping the actions in σ_2. Also, since all the derivation rules have just one judgment in their premise, we find it convenient to write them as rewrite rules: $\Gamma_1 \vdash \sigma_1 \longrightarrow \Gamma_2 \vdash \sigma_2$. Further, we let $\Longrightarrow$ denote the reflexive and transitive closure of $\longrightarrow$. Finally, we sometime write $\Gamma_1 \vdash \sigma_1 \overset{\Xi}{\Longrightarrow} \Gamma_n \vdash \sigma_n$ to state that Ξ is a (multistep) derivation from $\Gamma_1 \vdash \sigma_1$ to $\Gamma_n \vdash \sigma_n$.

Lemma 1 (Traces and Environments). *Let e be an event, and let $\Gamma \vdash \sigma$ with $e \in \Gamma$. Then $e \in \sigma$*

Proof. By induction on the derivation of $\Gamma \vdash \sigma$. The base case, when the judgement is derived by (EMPTY), follows vacuously. For the inductive case, let $\Gamma' \vdash \sigma' \longrightarrow \Gamma \vdash \sigma$ be the last step in the derivation. By induction hypothesis, we know that $e \in \sigma'$ for every $e \in \Gamma'$. We proceed by a case analysis on the rule applied in this step. First we observe that for all the derivation rules one has $\sigma = \sigma' :: \rho$ for a suitable trace ρ. Thus, the proof follows trivially for all the rules such that $\Gamma \subseteq \Gamma'$. In the only remaining case, (NEW), the proofs follows directly by an inspection of the rule. $\square$

Lemma 2 (Environments). *Assume $\Gamma \vdash \sigma$. Then Γ contains no pair of events built on the same (nonce or key) V.*

Proof. By induction on the derivation of $\Gamma \vdash \sigma$. The base case follows trivially, Γ being the empty set. For the inductive case, we proceed by a case analysis on the last rule in the derivation. If this rule is (NEW), $\Gamma \vdash \sigma$ is of the form $\Gamma', new(I,V) \vdash \sigma' :: new(I,V)$, with V fresh in σ'. By Lemma 1, we know that $new(I,V) \notin \Gamma'$. Then the proof follows by the induction hypothesis. $\square$

Lemma 3. *Assume $\Gamma_1, new(I,N) \vdash \sigma_1 \Longrightarrow \Gamma_2 \vdash \sigma_2 \overset{\Xi}{\Longrightarrow} \Gamma_3 \vdash \sigma_3$. If $new(I,N) \in \Gamma_3$ then $new(I,N) \in \Gamma_2$*

Proof. By induction on the number n of steps in Ξ. If $n = 0$ the proof follows trivially from the hypothesis, as $\Gamma_3 = \Gamma_2$. For the inductive case, let $\Gamma \vdash \sigma \longrightarrow \Gamma_3 \vdash \sigma_3$ be the last derivation step in Ξ. By Lemma1,we know that $new(I,N) \in \sigma_1$. Further, an inspection of the derivation rules shows that $\sigma_1 \sqsubseteq \sigma_2 \sqsubseteq \sigma$. Next, we analyze the transition $\Gamma \vdash \sigma \longrightarrow \Gamma_3 \vdash \sigma_3$. If $\Gamma_3 \subseteq \Gamma$ the proof follows by the induction hypothesis. If instead $\Gamma \subset \Gamma_3$, the transition must have been derived by (NEW). Hence $\Gamma_3 = \Gamma, new(I',V)$, and $\sigma_3 = \sigma :: new(I',V)$, with V fresh in σ. By Lemma 1, we thus know that $V \neq N$: then, $new(I,N) \in \Gamma_3$ implies $new(I,N) \in \Gamma$ and the proof follows again by the induction hypothesis. □

Lemma 4 (Nonce check). *Assume* $\Gamma \vdash \sigma_1 :: check(I,N) :: \sigma_2 :: check(I',N') :: \sigma_3$. *Then* $N \neq N'$.

Proof. An inspection of the deduction rules shows that the marker $check(I,N)$ is only introduced in a trace contextually to the event $new(I,N)$ being discarded from the environment. Hence, by Lemma 3, each nonce may be checked at most once in a trace. □

Lemma 5 (Nonces and Keys). *Assume* $\Gamma, new(I,V) \vdash \sigma_1 :: new(I,V) :: \sigma_2$. *Then* $\Gamma \vdash \sigma_1 :: \sigma_2$.

Proof. From the hypothesis, it follows that $\Gamma_1 \vdash \sigma_1$ for some Γ_1. Furthermore, an inspection of the derivation rules shows that the derivation of the judgment in the hypothesis must have the following form:

$$\Gamma_1 \vdash \sigma_1 \longrightarrow \Gamma_1, new(I,V) \vdash \sigma_1 :: new(I,V) \stackrel{\Xi}{\Longrightarrow} \Gamma, new(I,V) \vdash \sigma_1 :: new(I,V) :: \sigma_2$$

We prove the claim by induction on the number n of steps that it takes to generate σ_2 (i.e., on the length of the derivation Ξ). If $n = 0$, then $\sigma_2 = \varepsilon$, and the claim follows from our observation that $\Gamma_1 \vdash \sigma_1$ is derivable. For the inductive case, let

$$\Gamma', new(I,V) \vdash \sigma_1 :: new(I,V) :: \sigma_2' \longrightarrow \Gamma, new(I,V) \vdash \sigma_1 :: new(I,V) :: \sigma_2' :: \rho$$

denote the last step in Ξ (that the premise of this step has the format shown above follows from Lemma 3). By induction hypothesis, we know that $\Gamma' \vdash \sigma_1 :: \sigma_2'$. We need to show that $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$. We proceed by a case analysis on the derivation rule rule applied at the (last) step in question. We give only a few cases as representatives.

- Case (NEW) Then $\Gamma = \Gamma', new(I',V')$, and $\rho = new(I',V')$, with V' fresh in $\sigma_1 :: new(I,V) :: \sigma_2'$. But then V' is also fresh in $\sigma_1 :: \sigma_2'$: hence, from $\Gamma' \vdash \sigma_1 :: \sigma_2'$, by (NEW), we derive $\Gamma', new(I',V') \vdash \sigma_1 :: \sigma_2' :: \rho$, i.e. $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$, as desired.
- Case (PRINCIPLE 1) Then $\Gamma = \Gamma'$, $\rho = run_{\mathsf{Claim}} :: A \triangleright \langle \mathsf{Claim}, A, N \rangle_K$, and the only occurrences of $run_R(A,B)$ in $\sigma_1 :: new(I,V) :: \sigma_2'$ must have $R = \mathsf{Claim}$. Clearly, the same is true of the trace $\sigma_1 :: \sigma_2'$, and then $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$, derives from $\Gamma \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 1).
- Case(PRINCIPLE 2) Then $\Gamma = \Gamma'$, $\rho = run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}}$, and the only occurrences of $run_R(A,B)$ in $\sigma_1 :: new(I,V) :: \sigma_2'$ must have $R = \mathsf{Verif}$. The same is true in in the trace $\sigma_1 :: \sigma_2'$ and then $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$, derives from $\Gamma \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 2).
- Case(PRINCIPLE 3) Then $\Gamma = \Gamma' \setminus newS, N$ with $N \neq V$(by Lemma 4).$\rho is check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N' \rangle_{K_{BS}}$. Then clearly $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$ follows by an application of (PRINCIPLE 3).
- Case(PRINCIPLE 4) Then $\Gamma = \Gamma'$, $\rho = S \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{AS}}$. The reasoning doesn't differ if $N = V$ or $N \neq V$. $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$, derives from $\Gamma \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 4).
- Case(PRINCIPLE 5) Then $\Gamma = \Gamma' \setminus new(S,K)$ and obviously $K \neq V$. Then $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$, derives from $\Gamma' \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 5).

- Case(PRINCIPLE 6)Then $\Gamma = \Gamma'$, $\rho = run_{\mathsf{Owner}} :: A \triangleright \{M\}_K$ and $\Gamma \vdash \sigma_1 :: \sigma'_2 :: \rho$ derives from $\Gamma \vdash \sigma_1 :: \sigma'_2$ by an application of (PRINCIPLE 6).
- Case(PRINCIPLE 5') Then $\Gamma = \Gamma' \setminus new(S,N), new(S,K)$ with $N, K \neq V$ (by Lemma 4).ρ is $check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N', K \rangle_{K_{BS}}$. Then clearly $\Gamma \vdash \sigma_1 :: \sigma'_2 :: \rho$ follows by an application of (PRINCIPLE 5').
- Case(PRINCIPLE 5") Then $\Gamma = \Gamma' \setminus new(S,K)$ with $K \neq V$ (by Lemma 4).ρ is $S \triangleright \langle \mathsf{Claim}, A, N, K \rangle_{K_{BS}}$. Then clearly $\Gamma \vdash \sigma_1 :: \sigma'_2 :: \rho$ follows by an application of (PRINCIPLE 5").
- Case (AUTHENTICATION 1) Then $\rho = check(B,N) :: commit(B,A)$, and $\Gamma = \Gamma' \setminus \{new(B,N)\}$, with $N \neq V$. The proof follows by the induction hypothesis and an application of (AUTHENTICATION 1).
- Case(AUTHENTICATION 2) Then $\rho = check(B,N) :: commit(B,A)$, and $\Gamma = \Gamma' \setminus \{new(B,N)\}$, with $N \neq V$. The proof follows by the induction hypothesis and an application of (AUTHENTICATION 2).

The cases (PRINCIPLE 2), (PRINCIPLE 6) and (PRINCIPLE 4) follow exactly as case (PRINCIPLE 1). All the remaining cases follow as case (AUTHENTICATION 1). □

Lemma 6 (Claimant). *If* $\Gamma \vdash \sigma_1 :: run_{\mathsf{Claim}}(A,B) :: A \triangleright \langle \mathsf{Claim}, A, N \rangle_K :: \sigma_2$ *with* $new(I,N) \notin \Gamma$*, and* $check(J,N) \notin \sigma_2$*, then* $\Gamma \vdash \sigma_1 :: \sigma_2$.

Proof. By induction on the derivation of the judgment in the hypothesis. The proof is obvious by an inspection of the hypotheses of the rules and, for (AUTHENTICATION 1), remembering Lemma 4. □

Lemma 7 (Verifier). *If* $\Gamma \vdash \sigma_1 :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} :: \sigma_2$ *with* $new(I,N) \notin \Gamma$ *and* $check(J,N) \notin \sigma_2$*, then there exists* Γ^* *and* σ_2^* *such that* $\Gamma^* \vdash \sigma_1 :: \sigma_2^*$*, where* $\Gamma^* \subseteq \Gamma$*,* σ_2^* *is a subtrace of* σ_2*, and the following conditions are satisfied (for* $N \neq N'$*):*

- $I \triangleright \langle \mathsf{Claim}, I_1, N' \rangle_{K'} \in \sigma_2$ *implies* $I \triangleright \langle \mathsf{Claim}, I_1, N' \rangle_{K'} \in \sigma_2^*$
- $new(I,N') \in \Gamma$ *implies* $new(I,N') \in \Gamma^*$
- $A \triangleright \langle \mathsf{Verif}, B, N' \rangle_{K_{AS}} \in \sigma_2$ *implies* $A \triangleright \langle \mathsf{Verif}, B, N' \rangle_{K_{AS}} \in \sigma_2^*$
- $S \triangleright \langle \mathsf{Owner}, A, N'', K \rangle_{K_{BS}} \in \sigma_2 \setminus \sigma_2^*$ *implies* $S \triangleright \langle \mathsf{Owner}, B, N''', K \rangle_{K_{AS}} \notin \sigma_2^*$
- $\sigma_2 \setminus \sigma_2^*$ *is safe.*

Proof. From the hypothesis we know that $\Gamma_1 \vdash \sigma_1 :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}}$ for some Γ_1. We proceed by induction on the length of the derivation

$$\Gamma_1 \vdash \sigma_1 :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} \implies \Gamma \vdash \sigma_1 :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} :: \sigma_2$$

If the length is 0 then $\sigma_2^* = \varepsilon = \sigma_2$ and $\Gamma^* = \Gamma$ and the thesis holds. For the inductive case, let

$$\begin{aligned} \Gamma' \vdash \sigma_1 :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} :: \sigma'_2 \\ \longrightarrow \Gamma \vdash \sigma_1 :: run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} :: \sigma'_2 :: \rho \end{aligned}$$

be the last derivation step, with $\sigma_2 = \sigma'_2 :: \rho$. By induction hypothesis there exist $\Gamma'^* \subseteq \Gamma'$ and σ'^*_2 subtrace of σ'_2 such that $\Gamma'^* \vdash \sigma_1 :: \sigma'^*_2$ (and the conditions listed above are satisfied). We proceed by cases on the rule applied in the last step.

- Case (AUTHENTICATION 1) Then $\rho = check(B,N') :: commit(B,A)$ and $N' \neq N$. An inspection of the rule (AUTHENTICATION 1) shows that $new(B,N') \in \Gamma'$ and $S \triangleright \langle \mathsf{Claim}, A, N' \rangle_{K_{BS}} \in \sigma'_2$. By induction hypothesis, we know that $new(B,N') \in \Gamma'^*$ and $S \triangleright \langle \mathsf{Claim}, A, N' \rangle_{K_{BS}} \in \sigma'^*_2$. By an application of (AUTHENTICATION 1), we then have $\Gamma'^* \setminus \{new(B,N')\} \vdash \sigma_1 :: \sigma'^*_2 :: \rho$. Now the proof follows by taking $\Gamma^* = \Gamma'^* \setminus \{new(B,N')\}$ (and noting that $new(B,N') \notin \Gamma$).

- Case (AUTHENTICATION 2) Then ρ is $check(B,N') :: commit(B,A)$. If both $A \triangleright \{M\}_K$ and $S \triangleright \langle \mathsf{Owner}, A, N', K \rangle_{BS}$ are in $\sigma_1 :: \sigma_2'^*$ then $\Gamma \setminus \{new(B,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$ and the theorem holds. Now assume $S \triangleright \langle \mathsf{Owner}, A, N', K \rangle_{K_{BS}} \notin \sigma_1 :: \sigma_2'^*$. Then $S \triangleright \langle \mathsf{Owner}, B, N'', K \rangle_{K_{AS}} \notin \sigma_1 :: \sigma_2'^*$, hence also $run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K \notin \sigma_1 :: \sigma_2'^*$. Thus $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$ and, by Lemma 2, $\sigma_2 \setminus \sigma_2'^*$ is safe. If $A \triangleright \{M\}_K \notin \sigma_1 :: \sigma_2'^*$ then clearly $run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K \notin \sigma_1 :: \sigma_2'^*$. Thus $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$ and, by Lemma 2, $\sigma_2 \setminus \sigma_2'^*$ is safe.
- Case (NEW) Then ρ is $new(I,V)$. Of course V is fresh in $\sigma_1 :: \sigma_2'^*$. Then $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$.
- Case (PRINCIPLE 1) Then ρ is $run_{\mathsf{Claim}}(A,B) :: A \triangleright \langle \mathsf{Claim}, A, N' \rangle$. By an application of (PRINCIPLE 1) we have $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$. Obviously the theorem holds.
- Case (PRINCIPLE 2) Then ρ is $run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N' \rangle$. By an application of (PRINCIPLE 2) we have $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$.
- Case (PRINCIPLE 3) Then ρ is $check(S,N') :: S \triangleright \langle \mathsf{Claim}, A, N'' \rangle_{K_{BS}}$ and $N' \neq N$. An inspection of the rule proves $\Gamma'^* \setminus \{new(S,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$.
- Case (PRINCIPLE 4) Then ρ is $S \triangleright \langle \mathsf{Claim}, A, N' \rangle_{K_{BS}}$. If $N' \neq N$, then clearly $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$. Instead, if $N' = N$ one has $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$ and again the thesis of the theorem holds.
- Case(PRINCIPLE 5) Then ρ is $S \triangleright \langle \mathsf{Owner}, A, N', K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N'', K \rangle_{K_{AS}}$. $\Gamma \setminus new(S,K) \vdash \sigma_1 :: \sigma_2' :: \rho$, derives from $\Gamma'^* \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 5).
- Case (PRINCIPLE 5') Then ρ is $check(S,N') :: S \triangleright \langle \mathsf{Claim}, A, N'', K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N''', K \rangle_{K_{AS}}$ and $N' \neq N$. Then $\Gamma'^* \setminus \{new(S,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$ as desired.
- Case (PRINCIPLE 5") Then ρ is $S \triangleright \langle \mathsf{Claim}, A, N', K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N'', K \rangle_{K_{AS}}$. By hypothesis we have $new(S,K) \in \Gamma'^*$. If $N' \neq N$, then $\Gamma'^* \setminus \{new(S,K)\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$. Otherwise $\Gamma'^\vdash * \sigma_1 :: \sigma_2'^*$, and the theorem follows.
- Case(PRINCIPLE 6) Then ρ is $run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K$. If $S \triangleright \langle \mathsf{Owner}, B, N', K \rangle_{AS} \in \sigma_1 :: \sigma_2'^*$ then $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$. Otherwise $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$ and in both cases the theorem holds.

□

Lemma 8 (Owner). *If* $\Gamma \vdash \sigma_1 :: S \triangleright \langle \mathsf{Owner}, B, N', K' \rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N'', K' \rangle_{K_{BS}} :: \sigma_2$, $new(I,K') \notin \Gamma$, $new(I,N') \notin \Gamma$,$new(I,N'') \notin \Gamma$ *then there exist* Γ^* *and* σ_2^* *such that* $\Gamma^* \vdash \sigma_1 :: \sigma_2^*$ *, where* $\Gamma^* \subseteq \Gamma$,$\sigma_2^*$ *is a sublist of* σ_2. *In addition, the* Γ^* *and* σ_2^* *satisfy the following conditions (with* $N \neq N', N''$ *and* $K \neq K'$*):*

- $I \triangleright \langle \mathsf{Claim}, I_1, N \rangle_{K'} \in \sigma_2$ *implies* $I \triangleright \langle \mathsf{Claim}, I_1, N \rangle_{K'} \in \sigma_2^*$
- $new(I,N) \in \Gamma$ *implies* $new(I,N) in \Gamma^*$
- $A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} \in \sigma_2$ *implies* $A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}} \in \sigma_2^*$
- $new(A,K) \in \Gamma$ *implies* $new(A,K) \in \Gamma^*$
- $S \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{BS}} \in \sigma_2 \setminus \sigma_2^*$ $S \triangleright \langle \mathsf{Owner}, B, N''', K \rangle_{K_{AS}} \notin \sigma_2^*$
- $\sigma_2 \setminus \sigma_2^*$ *is safe.*

Proof. From the hypothesis we know that $\Gamma_1 \vdash \sigma_1 :: S \triangleright \langle \mathsf{Owner}, B, N', K' \rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N'', K' \rangle_{K_{BS}}$ for some Γ_1. We proceed by induction on the length of the derivation

$$\begin{aligned} &\Gamma_1 \vdash \sigma_1 :: S \triangleright \langle \mathsf{Owner}, B, N', K' \rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N'', K' \rangle_{K_{BS}} \\ &\qquad \Longrightarrow \Gamma \vdash \sigma_1 :: S \triangleright \langle \mathsf{Owner}, B, N', K' \rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N'', K' \rangle_{K_{BS}} :: \sigma_2 \end{aligned}$$

If the length is 0 then $\sigma_2^* = \varepsilon = \sigma_2$ and $\Gamma^* = \Gamma$ and the thesis holds. For the inductive case, let

$$\begin{aligned} &\Gamma' \vdash \sigma_1 :: S \triangleright \langle \mathsf{Owner}, B, N', K' \rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N'', K' \rangle_{K_{BS}} :: \sigma_2' \\ &\qquad \xrightarrow{R} \Gamma \vdash \sigma_1 :: S \triangleright \langle \mathsf{Owner}, B, N', K' \rangle_{K_{AS}} :: S \triangleright \langle \mathsf{Owner}, A, N'', K' \rangle_{K_{BS}} :: \sigma_2' :: \rho \end{aligned}$$

be the last rule in the derivation, for $\sigma_2 = \sigma_2' :: \rho$. By hypothesis $\exists \Gamma'^* \subseteq \Gamma'$ and $\sigma_2'^*$ sublist of σ_2' such that $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$. We reason on the last rule R:

- Case (NEW) Then ρ is $new(I,V)$. Of course V is fresh also in $\sigma_1 :: \sigma_2'^*$. Thus we have $new(I,V), \Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$. And this proves the thesis in this case.
- Case (PRINCIPLE 1) Then ρ is $run_{\mathsf{Claim}}(A,B) :: A \triangleright \langle \mathsf{Claim}, A, N \rangle$. We have $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$ by a simple application of (PRINCIPLE 1).
- Case (PRINCIPLE 2) Then ρ is $run_{\mathsf{Verif}}(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle$. By an application of (PRINCIPLE 2) we have $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$.
- Case (PRINCIPLE 3) Then ρ is $check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N''' \rangle_{K_{BS}}$ An inspection of the rule proves $\Gamma'^* \setminus \{new(S,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$.
- Case(PRINCIPLE 4) Then $\Gamma = \Gamma'$, $\rho = S \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{AS}}$. $\Gamma \vdash \sigma_1 :: \sigma_2' :: \rho$ derives from $\Gamma \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 4).
- Case(PRINCIPLE 5) Then ρ is $S \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N''', K \rangle_{K_{AS}}$. $\Gamma \setminus new(S,K) \vdash \sigma_1 :: \sigma_2' :: \rho$, derives from $\Gamma'^* \vdash \sigma_1 :: \sigma_2'$ by an application of (PRINCIPLE 5).
- Case(PRINCIPLE 5') Then ρ is $check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N'''', K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N''', K \rangle_{K_{AS}}$. A simple application of (PRINCIPLE 5') leads us to $\Gamma'^* \setminus \{new(S,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$ as desired.
- Case(PRINCIPLE 5") Then ρ is $S \triangleright \langle \mathsf{Claim}, A, N, K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N''', K \rangle_{K_{AS}}$. A simple application of (PRINCIPLE 5") leads us to $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$ as desired.
- Case(PRINCIPLE 6) Then ρ is $run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K$ By inductive hypothesis we don't know if $S \triangleright \langle \mathsf{Owner}, B, N, K \rangle_{AS} \in \sigma_1 :: \sigma_2'^*$. If it's so then $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^* :: \rho$, otherwise $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$. In both cases the theorem holds.
- Case (AUTHENTICATION 1) ρ is $check(B,N') :: commit(B,A)$. By hypotheses we have $J \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{BS}} \in sigma_1 :: \sigma_2'^*$. An application of (AUTHENTICATION 1) leads us to $\Gamma'^* \setminus \{new(B,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$.
- Case (AUTHENTICATION 2) Then ρ is $check(B,N') :: commit(B,A)$. If both $A \triangleright \{M\}_K$ and $S \triangleright \langle \mathsf{Owner}, A, N', K \rangle_{BS}$ are in $\sigma_1 :: \sigma_2'^*$ then $\Gamma'^* \setminus \{new(B,N')\} \vdash \sigma_1 :: \sigma_2'^* :: \rho$ and the theorem holds. Now, assume that $S \triangleright \langle \mathsf{Owner}, A, N', K \rangle_{BS} \notin \sigma_2'^*$. Then $S \triangleright \langle \mathsf{Owner}, B, N'', K \rangle_{K_{AS}} \notin \sigma_2'^*$, hence also $run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K \notin \sigma_2'^*$. Thus $\Gamma'^* \vdash \sigma_1 :: \sigma_2'^*$ and by Lemma 2 $\sigma_2 \setminus \sigma_2'^*$ is safe.
  If $A \triangleright \{M\}_K \notin \sigma_2'^*$ then obviously $run_{\mathsf{Owner}}(A,B) :: A \triangleright \{M\}_K \notin \sigma_2'^*$. Thus $\Gamma^* \vdash \sigma_1 :: \sigma_2'^*$ and, by Lemma 2, $\sigma_2 \setminus \sigma_2'^*$ is safe .

□

We are now ready to prove the safety theorem: we use the following, obvious, fact about safe traces.

Lemma 9. *If $\sigma :: \alpha :: \sigma'$ is safe then so is $\sigma :: \sigma'$ provided that $\alpha \neq run(A,B)$.*

Theorem 3 (Safety). *Assume $\Gamma \vdash \sigma$. Then σ is safe.*

Proof. By induction on the length of σ. The base case is when $\sigma = \varepsilon$, which is vacuously safe. For the inductive case, we only look at the case when σ is concluded by a commit action. Let then $\sigma = \sigma' :: check(B,N) :: commit(B,A)$. We now proceed by a case analysis on the last rule in the derivation of $\Gamma \vdash \sigma$. We only have two cases:

- Case (AUTHENTICATION 1). We further distinguish two sub-cases, depending on the premise of the rule.

  $I \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{AB}} \in \sigma'$. An inspection of the derivation rules shows that this action must have been generated by an application of (PRINCIPLE 1), in a derivation of the following shape (where

$\sigma_1' = \sigma_1 :: new(I,N)$ and $\sigma_2' = \sigma_2 :: run_{\mathsf{Claim}}(A,B) :: A \triangleright \langle \mathsf{Claim}, A, N\rangle_{K_{AB}}$).

$$\begin{aligned}
\Gamma_1 \vdash \sigma_1 &\longrightarrow \Gamma_1, new(I,N) \vdash \sigma_1 :: new(I,N)\\
&\Longrightarrow \Gamma_2 \vdash \sigma_1' :: \sigma_2\\
&\longrightarrow \Gamma_2 \vdash \sigma_1' :: \sigma_2 :: run_{\mathsf{Claim}}(A,B) :: A \triangleright \langle \mathsf{Claim}, A, N\rangle_{K_{AB}}\\
&\Longrightarrow \Gamma, new(I,N) \vdash \sigma_1' :: \sigma_2' :: \sigma_3\\
&\longrightarrow \Gamma \vdash \sigma_1' :: \sigma_2' :: \sigma_3 :: check(B,N) :: commit(B,A)
\end{aligned}$$

By Lemma 4, we know that N is not checked in σ_3. Then, from $\Gamma, new(I,N) \vdash \sigma_1' :: \sigma_2' :: \sigma_3$, by Lemma 5 and Lemma 6, we know that $\Gamma \vdash \sigma_1 :: \sigma_2 :: \sigma_3$ is derivable. Thus, by induction hypothesis, $\sigma_1 :: \sigma_2 :: \sigma_3$ is safe.

If instead $I \triangleright \langle \mathsf{Claim}, A, N\rangle_{K_{SB}} \in \sigma'$, then we need to consider several sub-cases, depending on the principle that generated $I \triangleright \langle \mathsf{Claim}, A, N\rangle_{K_{SB}}$.

- Case PRINCIPLE 3 . In the premises we have $I \triangleright \langle \mathsf{Verif}, B, N'\rangle_{K_{AS}} \in \sigma$", which must have been generated by PRINCIPLE 2 . Thus the derivation must have the folloiwng shape

$$\begin{aligned}
\Gamma_1 \vdash \sigma_1 &\rightarrow new(S,N), \Gamma_1 \vdash \overbrace{\sigma_1 :: new(S,N)}^{\sigma_1'}\\
&\Longrightarrow \Gamma_2 \vdash \sigma_1' :: \sigma_2\\
&\rightarrow \Gamma_2 \vdash \sigma_1' :: \overbrace{\sigma_2 :: run_V(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N\rangle_{K_{AS}}}^{\sigma_2'}\\
&\Longrightarrow \Gamma_3 \vdash \sigma_1' :: \sigma_2' :: \sigma_3\\
&\rightarrow new(B,N'), \Gamma_3 \vdash \sigma_1' :: \sigma_2' :: \overbrace{\sigma_3 :: new(B,N')}^{\sigma_3'}\\
&\Longrightarrow new(S,N), \Gamma_4 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4\\
&\rightarrow \Gamma_4 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \overbrace{\sigma_4 :: check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N'\rangle_{K_{BS}}}^{\sigma_4'}\\
&\Longrightarrow new(B,N'), \Gamma_5 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5\\
&\rightarrow \Gamma_5 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \overbrace{\sigma_5 :: check(B,N') :: commit(B,A)}^{\sigma_5'}
\end{aligned}$$

  We would like to show that $\sigma'' = \sigma_1 :: \sigma_2 :: \sigma_3 :: \sigma_4 :: \sigma_5$ is a trace. The reasoning is similar to previous case. σ_1 is a trace. Also $\sigma_1 :: \sigma_2$ is a trace by Lemma 5. By Lemma 7, there exists σ_3^* such that $\sigma_1 :: \sigma_2 :: \sigma_3^*$ is a trace. By Lemma 5 and Lemma 7 we can build a trace $sigma''^*$ subtrace of σ''. By inductive hypothesis σ''^* is safe and by construction so is σ'' is safe. Hence, σ also is safe.

- Case PRINCIPLE 4 . The derivation has the following shape:

$$
\begin{array}{rcl}
\Gamma_1 \vdash \sigma_1 & \to & new(B,N), \Gamma_1 \vdash \overbrace{\sigma_1 :: new(B,N)}^{\sigma'_1} \\
& \Longrightarrow & \Gamma_2 \vdash \sigma'_1 :: \sigma_2 \\
& \to & \Gamma_2 \vdash \sigma'_1 :: \overbrace{\sigma_2 :: run_V(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}}}^{\sigma'_2} \\
& \Longrightarrow & \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \sigma_3 \\
& \to & \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \overbrace{\sigma_3 :: S \triangleright \langle \mathsf{Claim}, A, N \rangle_{K_{BS}}}^{\sigma'_3} \\
& \Longrightarrow & new(B,N), \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma_4 \\
& \to & \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \overbrace{\sigma_4 :: check(B,N) :: commit(B,A)}^{\sigma'_4}
\end{array}
$$

Here we can drop the above listed actions out of σ by, respectively, Lemma 5, Lemma 7, Lemma 6. Thus also in this case σ is safe.

- Case PRINCIPLE 5' . The derivation has the following shape.

$$
\begin{array}{rcl}
\Gamma_1 \vdash \sigma_1 & \to & new(S,N), \Gamma_1 \vdash \overbrace{\sigma_1 :: new(S,N)}^{\sigma'_1} \\
& \Longrightarrow & \Gamma_2 \vdash \sigma'_1 :: \sigma_2 \\
& \to & \Gamma_2 \vdash \sigma'_1 :: \overbrace{\sigma_2 :: run_V(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}}}^{\sigma'_2} \\
& \Longrightarrow & \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \sigma_3 \\
& \to & new(B,N'), \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \overbrace{\sigma_3 :: new(B,N')}^{\sigma'_3} \\
& \Longrightarrow & \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma_4 \\
& \to & new(S,K), \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \overbrace{\sigma_4 :: new(S,K)}^{\sigma'_4} \\
& \Longrightarrow & new(S,N), new(S,K), \Gamma_5 \vdash \overbrace{\sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4}^{\sigma_{1234}} :: \sigma_5 \\
& \to & \Gamma_5 \vdash \sigma'_{1234} :: \overbrace{\sigma_5 :: check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N', K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N'', K \rangle_{K_{AS}}}^{\sigma'_5} \\
& \Longrightarrow & new(B,N'), \Gamma_6 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4 :: \sigma'_5 :: \sigma_6 \\
& \to & \Gamma_6 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4 :: \sigma'_5 :: \overbrace{\sigma_6 :: check(B,N') :: commit(B,A)}^{\sigma'_6}
\end{array}
$$

In order to discard the above listed actions from σ, one can use, respectively, Lemma 5, Lemma 7, Lemma 5 again, Lemma 5 and finally Lemma 8 and Lemma 6.

- Case PRINCIPLE 5" . The derivation has the following shape:

$$
\begin{aligned}
\Gamma_1 \vdash \sigma_1 \;&\rightarrow\; new(B,N), \Gamma_1 \vdash \overbrace{\sigma_1 :: new(B,N)}^{\sigma'_1} \\
&\Longrightarrow\; \Gamma_2 \vdash \sigma'_1 :: \sigma_2 \\
&\rightarrow\; \Gamma_2 \vdash \sigma'_1 :: \overbrace{\sigma_2 :: run_V(A,B) :: A \triangleright \langle \mathsf{Verif}, B, N \rangle_{K_{AS}}}^{\sigma'_2} \\
&\Longrightarrow\; \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \sigma_3 \\
&\rightarrow\; new(S,K), \Gamma_3 \vdash \overbrace{\sigma'_1, \sigma'_2, \sigma_3 :: new(S,K)}^{\sigma'_3} \\
&\Longrightarrow\; new(S,K), \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma_4 \\
&\rightarrow\; \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \overbrace{\sigma_4 :: S \triangleright \langle \mathsf{Claim}, A, N, K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N', K \rangle_{K_{AS}}}^{\sigma'_4}. \\
&\Longrightarrow\; new(B,N), \Gamma_5 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4 :: \sigma_5 \\
&\rightarrow\; \Gamma_5 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4 :: \overbrace{\sigma_5 :: check(B,N) :: commit(B,A)}^{\sigma'_4}
\end{aligned}
$$

As usual, you can drop the above listed actions by Lemma 5, Lemma 7, Lemma 5 and finally Lemma 7(Case PRINCIPLE 5").

– <u>Case (AUTHENTICATION 2)</u> We distinguish several subcases, depending on the premises of the rule $I \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{SB}} \in \sigma'$ and $A \triangleright \{M\}_K \in \sigma'$.

- Let $A \triangleright \{M\}_K$ be generated by PRINCIPLE 6 and $I \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{SB}}$ by PRINCIPLE 5 . The derivation has the following shape:

$$
\begin{aligned}
\Gamma_1 \vdash \sigma_1 \;&\rightarrow\; new(S,K), \Gamma_1 \vdash \overbrace{\sigma_1 :: new(S,K)}^{\sigma'_1} \\
&\Longrightarrow\; \Gamma_2 \vdash \sigma'_1 :: \sigma_2 \\
&\rightarrow\; new(B,N), \Gamma_2 \vdash \sigma'_1 :: \overbrace{\sigma_2 :: new(B,N)}^{\sigma'_2} \\
&\Longrightarrow\; new(S,K), \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \sigma_3 \\
&\rightarrow\; \Gamma_3 \vdash \sigma'_1 :: \sigma'_2 :: \overbrace{\sigma_3 :: S \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N', K \rangle_{K_{AS}}}^{\sigma'_3} \\
&\Longrightarrow\; \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma_4 \\
&\rightarrow\; \Gamma_4 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \overbrace{\sigma_4 :: run_K(A,B) :: A \triangleright \{M'\}_K}^{\sigma'_4} \\
&\Longrightarrow\; new(B,N), \Gamma_5 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4 :: \sigma_5 \\
&\rightarrow\; \Gamma_5 \vdash \sigma'_1 :: \sigma'_2 :: \sigma'_3 :: \sigma'_4 :: \overbrace{\sigma_5 :: check(B,N) :: commit(B,A)}^{\sigma'_5}
\end{aligned}
$$

This time to discard the above listed actions from σ, we use Lemma 5 twice, Lemma 8 and finally Lemma 8 (Case PRINCIPLE 6).

- Let $A \triangleright \{M\}_K$ be generated by PRINCIPLE 6 and $I \triangleright \langle \mathsf{Owner}, A, N, K \rangle_{K_{SB}}$ by PRINCIPLE 5' . The derivation has the following shape:

$$
\begin{aligned}
\Gamma_1 \vdash \sigma_1 \;&\rightarrow\; new(S,N),\Gamma_1 \vdash \overbrace{\sigma_1 :: new(S,N)}^{\sigma_1'}\\
&\Longrightarrow\; \Gamma_2 \vdash \sigma_1' :: \sigma_2\\
&\rightarrow\; \Gamma_2 \vdash \sigma_1' :: \overbrace{\sigma_2 :: run_V(B,A) :: B \triangleright \langle \mathsf{Verif}, A, N\rangle_{K_{BS}}}^{\sigma_2'}\\
&\Longrightarrow\; \Gamma_3 \vdash \sigma_1' :: \sigma_2' :: \sigma_3\\
&\rightarrow\; new(B,N''),\Gamma_3 \vdash \sigma_1' :: \sigma_2' :: \overbrace{\sigma_3 :: new(B,N'')}^{\sigma_3'}\\
&\Longrightarrow\; \Gamma_4 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4\\
&\rightarrow\; new(S,K),\Gamma_4 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \overbrace{\sigma_4 :: new(S,K)}^{\sigma_4'}\\
&\Longrightarrow\; \Gamma_5 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5\\
&\rightarrow\; new(A,N'),\Gamma_5 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \overbrace{\sigma_5 :: new(A,N')}^{\sigma_5'}\\
&\Longrightarrow\; new(S,N),\Gamma_6 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6\\
&\rightarrow\; \Gamma_6 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \overbrace{\sigma_6 :: check(S,N) :: S \triangleright \langle \mathsf{Claim}, A, N'', K\rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N', K\rangle_{K_{AS}}}^{\sigma_6'}\\
&\Longrightarrow\; \Gamma_7 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6' :: \sigma_7\\
&\rightarrow\; \Gamma_7 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6' :: \overbrace{\sigma_7 :: run_K(A,B) :: A \triangleright \{M\}_K}^{\sigma_7'}\\
&\Longrightarrow\; new(B,N''),\Gamma_8 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6' :: \sigma_7' :: \sigma_8\\
&\rightarrow\; \Gamma_8 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6' :: \sigma_7' :: \overbrace{\sigma_8 :: check(B,N'') :: commit(B,A)}^{\sigma_8'}
\end{aligned}
$$

Here we can drop the displayed actions out of σ by using Lemma 5, Lemma 7, Lemma 5 three times, 6 and 8.

- Let $A \triangleright \{M\}_K$ be generated by PRINCIPLE 6 and $I \triangleright \langle \mathsf{Owner}, A, N, K\rangle_{K_{SB}}$ by PRINCIPLE 5" . The derivation has the following shape:

$$
\begin{aligned}
\Gamma_1 \vdash \sigma_1 \;&\rightarrow\; new(B,N),\Gamma_1 \vdash \overbrace{\sigma_1 :: new(B,N)}^{\sigma_1'}\\
&\Longrightarrow\; \Gamma_2 \vdash \sigma_1' :: \sigma_2\\
&\rightarrow\; \Gamma_2 \vdash \sigma_1' :: \overbrace{\sigma_2 :: run_V(B,A) :: B \triangleright \langle \mathsf{Verif}, A, N\rangle_{K_{BS}}}^{\sigma_2'}\\
&\Longrightarrow\; \Gamma_3 \vdash \sigma_1' :: \sigma_2' :: \sigma_3\\
&\rightarrow\; new(S,K),\Gamma_3 \vdash \sigma_1' :: \sigma_2' :: \overbrace{\sigma_3 :: new(S,K)}^{\sigma_3'}\\
&\Longrightarrow\; \Gamma_4 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4\\
&\rightarrow\; new(A,N'),\Gamma_4 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \overbrace{\sigma_4 :: new(A,N')}^{\sigma_4'}\\
&\Longrightarrow\; \Gamma_5 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5\\
&\rightarrow\; \Gamma_5 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \overbrace{\sigma_5 :: S \triangleright \langle \mathsf{Claim}, A, N, K\rangle_{K_{BS}} :: S \triangleright \langle \mathsf{Owner}, B, N', K\rangle_{K_{AS}}}^{\sigma_5'}\\
&\Longrightarrow\; \Gamma_6 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6\\
&\rightarrow\; \Gamma_6 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \overbrace{\sigma_6 :: run_K(A,B) :: A \triangleright \{M\}_K}^{\sigma_6'}\\
&\Longrightarrow\; new(B,N''),\Gamma_7 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6' :: \sigma_7\\
&\rightarrow\; \Gamma_7 \vdash \sigma_1' :: \sigma_2' :: \sigma_3' :: \sigma_4' :: \sigma_5' :: \sigma_6' :: \overbrace{\sigma_7 :: check(B,N) :: commit(B,A)}^{\sigma_7'}
\end{aligned}
$$

α is safe and the reasoning is clearly the same as previous case and it uses Lemma 7(Case PRINCIPLE 5").