ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Technical Report No. 528

Formalizing and Analyzing Sender Invariance*

Paul Hankes Drielsma, Sebastian Mödersheim, Luca Viganò, David Basin

Information Security Group, Dep. of Computer Science, ETH Zurich, Switzerland
www.infsec.ethz.ch/~{drielsma,moedersheim,vigano,basin}
{drielsma,moedersheim,vigano,basin}@inf.ethz.ch

Abstract. In many network applications and services, agents that share no secure channel in advance may still wish to communicate securely with each other. In such settings, one often settles for achieving security goals weaker than authentication, such as sender invariance. Informally, sender invariance means that all messages that seem to come from the same source actually do, where the source can perhaps only be identified by a pseudonym. This implies, in particular, that the relevant parts of messages cannot be modified by an intruder.
In this paper, we provide the first formal definition of sender invariance as well as a stronger security goal that we call strong sender invariance. We show that both kinds of sender invariance are closely related to, and entailed by, weak authentication, the primary difference being that sender invariance is designed for the context where agents can only be identified pseudonymously. In addition to clarifying how sender invariance and authentication are related, this result shows how a broad class of automated tools can be used for the analysis of sender invariance protocols. As a case study, we describe the analysis of two sender invariance protocols using the OFMC back-end of the AVISPA Tool.

1 Introduction

The establishment of a secure channel between communicating parties requires a pre-existing relationship between them. Examples of such relationships include shared passwords and transitive relationships, for instance where the parties exchange public-key certificates issued by a trusted certification authority. Common to these types of relationships is the prerequisite that some data is available in order to bootstrap the secure channel. This data may have been exchanged in

* This work was partially supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322, and by the Zurich Information Security Center. It represents the views of the authors.

advance, or it might be produced on the fly by a reliable source like a certification authority.

In many network applications and services, however, agents that share no such bootstrap data in advance may still wish to communicate securely with each other. Indeed, they might have no prior relationship whatsoever, not even a transitive one. In such situations, classical authentication (e.g. via public keys) is impossible. To remedy this problem, one would have to define a process by which bootstrap data is established and distributed, but in many settings this is too expensive or cumbersome, as one must require that every participant must somehow "register" before using a service. For a small coffee shop offering its customers a wireless hotspot, such a registration process could detract from one of its main selling points: convenience.

In settings where no bootstrap data is available, a weaker security goal is still achievable, namely *sender invariance*. Informally, sender invariance means that all messages that seem to come from the same source actually do, where the source can perhaps only be identified by a pseudonym. This implies, in particular, that the messages cannot be modified by an intruder, at least not their relevant parts. Moreover, one may want to ensure the secrecy of the communicated data between the participants.

Sender invariance arises in a variety of situations, both wired and wireless. Consider, for instance, any of the many free web-based e-mail services available online. In general, users register for an online e-mail service via an informal process whereby a new username and password are established. At no point does a formal authentication process take place involving, for example, photo identification or a physical signature. This process is acceptable, as an email address can be seen as a pseudonym that is not necessarily linkable with its owner's true identity. This, however, has ramifications concerning how one should describe the login process, as there is no reliable means of linking the established username with the identity of the user. If one considers the login process and ignores registration, simply assuming that credentials have been exchanged sometime in the past, then the user login process can be called authentication. However, in light of this informal registration process, the login process should more accurately be described as ensuring sender invariance: that is, the user with pseudonym *John Doe* is the same user who originally opened the account registered as *John Doe*, although the e-mail provider does not know his proper identity.

Such online services, whose users can be logged in based only on credentials which are not linkable to their actual identities, are already prevalent. As networks move towards increased mobility and ad-hoc connections, situations in which reliable pre-shared cryptographic credentials are limited or unavailable will arise with increasing frequency. Understanding what security goals can be achieved in such situations is important for the design of next-generation network protocols. In this paper, we aim to further this understanding by examining sender invariance in detail.

Contributions: Our first contribution is a formal definition of sender invariance and a stronger, related security goal which we call *strong sender invariance*. We also show that (strong) sender invariance is closely related to weak authentication, the primary difference being that sender invariance assumes that agents can only be identified pseudonymously. Based on this, we show that the three security goals constitute a hierarchy. Furthermore, we show how a broad class of automated analysis tools can be used to analyze sender invariance protocols. We also describe the analysis of two protocols using the On-the-Fly Model Checker OFMC [7], one of the back-ends of the AVISPA Tool for security protocol analysis [3].

Related Work: Our work focuses on the formal definition security goals and the relationships between them. Gollmann [14] considered authentication in detail, and Lowe [16] subsequently defined a hierarchy of authentication goals which apply in settings where relationships between agents have been established in advance. In §2.5, we similarly define a hierarchy that relates the two forms of sender invariance and weak authentication. Our focus, however, is on settings in which the kinds of relationships that one would normally require to bootstrap a secure channel are not available.

Our motivation to examine settings where agents know each other perhaps only via pseudonyms was inspired by current trends in protocol development, in particular the work of the Internet Engineering Task Force on Mobile IPv6 [15]. In [17], for instance, the authors identify sender invariance as a goal that should be ensured by the IPv6 SEcure Neighbor Discovery protocol (SEND [2]) in ad hoc networks.[1] In SEND, this is achieved via a mechanism for providing sender invariance called Cryptographically Generated Addresses (CGA [4]). In this paper, we consider a similar idea, the Purpose-Built Keys Framework [10].

Organization: In §2, we define and discuss sender invariance and strong sender invariance. In §3, we present a case study based on the Purpose-Built Keys Framework that illustrates the formal analysis of sender invariance with the OFMC tool. In §4, we discuss settings in which agents share some bootstrap data, but not enough to achieve mutual authentication. In §5, we summarize our results and discuss future work.

2 Sender Invariance

Designers of modern security protocols face a challenge. On the one hand, protocols need to be designed with ever-increasing mobility in mind. On the other hand, this very mobility means that designers should also make few assumptions about the amount of information shared, in advance, among protocol participants; indeed, one must often assume that participants share no a priori relationships at all. Yet authentication protocols tend to rely on just such pre-shared information, such as a public key or a shared password. Indeed, in [9],

[1] Note that the authors do not actually call the goal sender invariance, but merely describe the intuition: "nodes ensure that they are talking to the same nodes (as before)" [17, §3.3].

Boyd argues that in the absence of authenticated shared information, no secure channels can be established.

Sender invariance protocols are based on the idea that, in many situations, one party of a protocol does not need to be authenticated in the classical sense, but rather could pick a pseudonym and be identified by that pseudonym thereafter. The protocols ensure that an intruder cannot "take over" somebody else's pseudonym, i.e. generate messages that appear to originate from the owner of the pseudonym, or read messages that are sent to the owner of the pseudonym.

A variety of mechanisms can be used to realize sender invariance. Perhaps the most common one, and the one used in our running example PBK, is as follows. An agent creates an asymmetric key pair, publishes the public key, and uses a hash value of the public key as a pseudonym. Clearly, the intruder can generate his own pseudonym, but he cannot sign or decrypt messages with the private key associated with somebody else's pseudonym. The remarkable thing about these mechanisms is thus that we get—out of nothing—variants of authentic channels that only differ from the classical ones by the fact that one end point is identified by a pseudonym.

The goals that are considered for sender invariance protocols are thus similar to classical authentication and secrecy goals, but with the twist that one side is identified by a pseudonym rather than a real name. By *sender invariance*, we informally mean that all messages come from the same source that is identified by a pseudonym:

> A two-party protocol P guarantees the responder role *sender invariance with respect to the initiator role* iff the following holds: whenever an agent b in the responder role receives a message that appears to have been sent by an agent with pseudonym id, then this message originates from the same agent playing the initiator role as all previous messages that appeared to come from pseudonym id.

Note that sender invariance differs in several respects from privacy (for instance, the privacy properties defined in [1]). Privacy means to protect the identities of the communicating agents from being observable (to an outstanding party or even to each other); for sender invariance, the protection of identities is not an issue (and agents may expose their identities, even if they cannot prove them). Sender invariance is rather the best we can achieve when identification/authentication is not possible.

The relation of this goal with classical authentication will be discussed shortly. We note that one may similarly develop a concept of *receiver invariance* as an analogue of secrecy goals in this pseudonym-based communication; we do not, however, consider this further in this paper.

2.1 Purpose-Built Keys

As a running example, we introduce a protocol based on the Purpose-Built Keys Framework (PBK [10]), a mechanism for achieving sender invariance. PBK uses

1. $A \rightarrow B : PBK_A$
....
2. $A \rightarrow B : \{Msg\}_{{PBK_A}^{-1}}.H(PBK_A)$
3. $B \rightarrow A : N_B.H(PBK_A)$
4. $A \rightarrow B : \{N_B\}_{{PBK_A}^{-1}}.H(PBK_A)$

Protocol 1: An example PBK Protocol

freshly generated, temporary, asymmetric key pairs. A user's pseudonym is simply a hash of the temporary public key, the so-called PBID. In an initialization phase, the sender agent transmits his purpose-built public key. If this exchange is not tampered with, then the sender can sign subsequent messages, thus assuring the receiver that the source of the messages has not changed.

We note that denial of service attacks are possible, in the sense that the intruder can drop messages from an honest initiator. We do not consider such attacks here, however, as they do not constitute violations of sender invariance.

Example. Protocol 1 is an example protocol which uses PBK to ensure sender invariance between an initiator A and a responder B. Upon starting the protocol, A generates her purpose-built key pair PBK_A and ${PBK_A}^{-1}$. She sends the former to B in message 1. The dotted line separates the initialization phase from the rest of the protocol. In message 2, A sends some payload Msg to B signed with her PBK. Messages 3 and 4 perform a challenge-response exchange in order to prove to B that the party purporting to possess ${PBK_A}^{-1}$ is indeed active and the signed messages are not simply being replayed. We assume that A and B might want to exchange multiple payload messages with the pseudonym $H(PBK_A)$, so messages 2 through 4 might be repeated arbitrarily often.

The running example of Protocol 1 will serve as a basis for the discussions below, where we describe our model and define sender invariance formally.

2.2 Formalizing Sender Invariance

The informal definition given above is meant to provide the intuition behind the goal of sender invariance: namely, that a sequence of messages that apparently all originate from the same sender truly do. Note that we do not assume that the agent playing the responder role knows the real identity of the initiator with whom he communicates; this property should hold even if the receiver knows the sender only via some pseudonym. It is this intuition that we strive to capture in our formal definition of sender invariance below.

To formulate sender invariance independently of which particular formalism or tool is adopted for modeling and analysis, we define requirements for protocol models (summarized in Fig. 1), that are sufficient to formalize sender invariance. We assume that there exists a set Msg of all *messages*, which we represent as free terms with the standard perfect cryptography assumption. Let $\mathsf{Agent} \subseteq \mathsf{Msg}$

E	A set of events
$\mathsf{AE} \supseteq \{witness, request\}$	Authentication events, with $\mathsf{AE} \subseteq \mathsf{E}$
Msg	The set of all possible messages
$\mathsf{Agent} \subseteq \mathsf{Msg}$	Agent identifiers, both names and pseudonyms
$\mathsf{HAgent} \subseteq \mathsf{Agent}$	Honest agent identifiers
$\mathsf{ID} \subseteq \mathsf{Agent}$	Pseudonyms
$\mathsf{H\_ID} = \mathsf{ID} \cap \mathsf{HAgent}$	Pseudonyms belonging to honest agents
Vars	Set of protocol variable identifiers

Fig. 1. Notation

denote the set of all possible *agent identifiers*, including both real names and a set $\mathsf{ID} \subseteq \mathsf{Agent}$ of *pseudonyms*. We also assume that there exists a set of *honest agent identifiers*, which we denote $\mathsf{HAgent} \subseteq \mathsf{Agent}$, and a set of *honest pseudonyms*, which is $\mathsf{H\_ID} = \mathsf{ID} \cap \mathsf{HAgent}$. As notation, we will use upper case $A, B, ...$ to denote role names and lower case $a, b, ...$ for agent names.

We follow the standard Dolev-Yao model [13] of an active intruder who controls the network but cannot break cryptography: the intruder can intercept messages and analyze them if he possesses the respective keys for decryption, and he can generate messages from his knowledge and send them under any party's name.

The protocol models must also provide some way to reason about the way that an agent interprets particular concrete messages. In Protocol 1, for instance, the responder B might want to ensure that a concrete value he receives and interprets as A's payload message Msg was indeed intended by A as a payload message and not, for instance, as a response to the challenge N_B. To this end, we require the existence of a set Vars of identifiers for the *variables of the protocol*. The elements of Vars are logical identifiers indicating how an agent interprets a given value. The definition of the set itself is protocol specific. For instance, for Protocol 1, the set $\mathsf{Vars} = \{PBK_A, Msg, N_B\}$ would be appropriate.

We assume that protocol models have behaviors that can be expressed as *linearly ordered traces of events* from a fixed event set E. Traces contain events from a set AE of *auxiliary events* that express information about an honest agent's assumptions or intentions when executing a protocol. These events provide a language over which we then define the goals of the protocol.[2] We assume that the intruder can neither generate events from AE nor modify those AE events generated by honest agents. By convention, we call the events in AE *witness* and *request*. For $a, b \in \mathsf{Agent}$, $v \in \mathsf{Vars}$, and $m \in \mathsf{Msg}$,

[2] This approach to formalizing protocol goals is standard. It is adopted, for instance, in the AVISPA Tool [3, 7], and it is analogous to other approaches like that of [16], where goals are formulated in terms of "status signals" exchanged on special channels to which the intruder has no access.

- *witness*(a, b, v, m) expresses that initiator a intends to execute the protocol with responder b and wishes to use value m as the protocol variable v; and
- *request*(b, a, v, m) expresses that responder b accepts the value m and now relies on the guarantee that agent a exists and agrees with him on this value for protocol variable v.

Consider an honest initiator a who wishes to execute a protocol with a responder b. For all $v \in \mathsf{Vars}$ that are of interest (where the definition of interest will depend strongly on the goals of the protocol in question), a will generate an event *witness*(a, b, v, m) upon setting a value m for v, and each honest responder will generate an event *request*(b, a, v, m') after reaching an accepting state in which he has assigned the value m' to v. Following [3, 7], we define protocol goals below as conditions on traces that specify how *witness* and *request* events must correspond with one another.

Example. In Protocol 1, one can define the variables of interest to be those which the responder wants to be sure originated from the pseudonym $H(PBK_A)$: namely, *Msg* and the signed N_B. Honest agents will, as mentioned, generate auxiliary events for each of these variables of interest, but we consider only *Msg* in this example. We assume that agent a with PBK pbk_a wishes to execute Protocol 1 with agent b, and that a wishes to transmit the payload message 17. Furthermore, for the sake of example, we ignore possible manipulations by the intruder and assume that messages are transmitted without modification.

Upon sending message 1, a generates the event $witness(H(pbk_a), b, Msg, 17)$, expressing that, under her pseudonym, she intends to send to b the value 17, interpreting it as protocol variable *Msg*. The responder accepts the protocol run only after receiving message 4, which confirms recentness. After receiving message 4, b will generate the event $request(b, H(pbk_a), Msg, 17)$, indicating that he accepts the value 17, believes that it originates from the agent associated with pseudonym $H(pbk_a)$, and interprets it as the protocol variable *Msg*.

We now formally define the security goal of sender invariance as the following temporal property of traces of events over the set E, where $\Box$ and $\diamondsuit\!\!\!\!-$ denote the linear time temporal operators "always in the future" and "sometime in the past", respectively:

$$\begin{aligned}\textsc{si}:\ & \forall b \in \mathsf{HAgent}.\forall id \in \mathsf{H\_ID}.\forall m \in \mathsf{Msg}.\forall v \in \mathsf{Vars}.\\ & \quad \Box(request(b, id, v, m) \rightarrow \exists v' \in \mathsf{Vars}.\,\diamondsuit\!\!\!\!-\, witness(id, b, v', m))\end{aligned}$$

We assume, in this definition, that the initiator knows the real name of the responder b, but we do not require that b knows the real name of the initiator. This definition expresses that every honest agent b is guaranteed that, if $id \in \mathsf{H\_ID}$, then there exists an honest agent who sent all the values m that b believes originated from pseudonym id. Recall that only honest agents generate the auxiliary events in AE, therefore the presence of a *witness* event implies that it was generated by an honest agent. Moreover, for each incoming message m that b associates with the protocol variable v in the *request*, there exists some protocol variable v' that expresses how the honest owner of pseudonym id intended to send the value m. This implies that the values m have not been modified in transit, but the sender and receiver may have assigned different interpretations to the transmitted values.

2.3 Strong Sender Invariance

A stronger goal results when the interpretations must agree. We define *strong sender invariance*, a modification of sender invariance, by requiring that the sender and the receiver agree on the interpretation of each message. We formalize this as follows:

$$\begin{aligned}\textsc{strongsi}:\ & \forall b \in \mathsf{HAgent}.\forall id \in \mathsf{H\_ID}.\forall m \in \mathsf{Msg}.\forall v \in \mathsf{Vars}.\\ & \Box(\mathit{request}(b, id, v, m) \rightarrow \Diamond\!\!\!\!-\, \mathit{witness}(id, b, v, m))\end{aligned}$$

Strong sender invariance, as the name implies, provides a stronger guarantee than sender invariance itself (we will show this formally in §2.5). Specifically, it requires that all values m received by b apparently from an honest pseudonym id indeed originated from the same honest agent. Moreover, for each m, the protocol variable v with which b associates m must be the same as the v for which the value was intended by the sender id; that is, v is the same in both auxiliary events. As before, this implies that the value was not modified in transit, but we now additionally require that the interpretations agree. In the extreme case, that the protocol-specific set of "interesting" protocol variables includes *all* protocol variables, this implies that the exact messages sent by the initiator arrive, without tampering, at the responder.

2.4 Discussion

The informal notion that the source of a communication does not change suffers from ambiguities that one must resolve when defining sender invariance formally. Perhaps most importantly, one must define to what extent sender invariance implies message integrity.

Conservatively, one can define sender invariance in such a way that any message modification violates sender invariance. This would be akin to the notion of matching conversations, defined in [8]. Such a definition is quite restrictive and of limited practical use, particularly in ad-hoc settings with potentially no relationships among protocol participants.

Instead, we opt for a finer-grained definition in which integrity must be guaranteed only for relevant parts of the exchanged messages, where "relevant" can be defined in a protocol-specific way. The case described above is then a special case of this more general approach in which all parts of the protocol messages are considered relevant. In order to pursue this fine-grained approach, we formalize sender invariance over the auxiliary trace events *witness* and *request* rather than, for instance, over the communication events themselves.

The auxiliary events *witness* and *request* confer a further benefit; namely, they contain all the information one needs to formalize authentication itself.

This facilitates a direct comparison of the two forms of sender invariance with authentication, discussed in the next subsection.

Finally, we note that alternate definitions of (strong) sender invariance are also possible and may be appropriate for certain settings. In our definition, we assume a setting in which the owner of pseudonym *id* knows the identity of the agent b with whom he wants to communicate. This assumption is appropriate for one of our larger case-study protocols, Mobile IPv6 [15]. One could, however, envision protocols in which the recipient is unimportant, or indeed known via a pseudonym. For such protocols, one might define sender invariance as follows (and strong sender invariance analogously):

$$\begin{aligned}\textsc{si}'\colon\ & \forall b \in \mathsf{HAgent}.\forall id \in \mathsf{H\_ID}.\forall m \in \mathsf{Msg}.\forall v \in \mathsf{Vars}.\exists b' \in \mathsf{Agent}.\\ & \qquad \Box(request(b, id, v, m) \rightarrow \exists v' \in \mathsf{Vars}.\,\diamondminus\, witness(id, b', v', m))\ .\end{aligned}$$

For the rest of the paper, however, we will focus on our original definition SI.

2.5 Relating Sender Invariance and Authentication

We now examine the relationship between sender invariance, strong sender invariance, and authentication. We first recall the informal definition of weak authentication (adapted from [16], where it is termed non-injective agreement):

> A protocol guarantees *weak authentication* to a responder B on a set of protocol variables V iff whenever B completes a run of the protocol, apparently with initiator A, then A has previously been executing the protocol as initiator, apparently with responder B, and the two agents agree on the data values corresponding to all the variables in V.

In our model, we equate the responder's completion of a protocol run with his arrival in an accepting state. Since we assume that responders issue *request* events only after reaching an accepting state, we can formally define weak authentication as follows:

$$\begin{aligned}\textsc{wauth}\colon\ & \forall b \in \mathsf{HAgent}.\forall a \in \mathsf{HAgent}.\forall m \in \mathsf{Msg}.\forall v \in \mathsf{Vars}.\\ & \Box(request(b, a, v, m) \rightarrow \diamondminus\, witness(a, b, v, m))\end{aligned}$$

Observe that strong sender invariance differs from weak authentication only in the inclusion of the pseudonym $id \in \mathsf{H\_ID}$ rather than an actual agent identifier $b \in \mathsf{HAgent}$, which may be either a pseudonym or a real name. Thus, strong sender invariance is the direct analogue to weak authentication for the pseudonymous setting, and we have that WAUTH implies STRONGSI. The converse, however, does not hold, as expressed as Proposition 1.

Proposition 1 *Weak authentication is a strictly stronger security goal than strong sender invariance.*

Proof. We first show that every trace that satisfies weak authentication also satisfies strong sender invariance; thus, if all traces induced by a protocol satisfy weak authentication, then they also satisfy strong sender invariance. To that end, consider an arbitrary trace that satisfies weak authentication and any event on this trace of the form $request(b, id, v, m)$, for arbitrary $b \in \mathsf{HAgent}$, $id \in \mathsf{H\_ID}$, $v \in \mathsf{Vars}$ and $m \in \mathsf{Msg}$. We have to show that this event is preceded by the event $witness(id, b, v, m)$. This follows directly, since $\mathsf{H\_ID} \subseteq \mathsf{HAgent}$, and weak authentication demands that any event $request(b, a, v, m)$—where now $a \in \mathsf{HAgent}$—is preceded by $witness(a, b, v, m)$. Note that if $a \notin \mathsf{H\_ID}$ for all initiators $a \in \mathsf{HAgent}$ for which request terms are generated, then sender invariance holds trivially. Since we have not assumed any specific property about b, id, v, and m, or where in the trace the *request* event occurs, every $request(b, id, v, m)$ is preceded by $witness(id, b, v, m)$.

To see that weak authentication is strictly stronger than strong sender invariance, consider a trace with the event $request(b, a, v, m)$ with $a \in \mathsf{HAgent} \setminus \mathsf{H\_ID}$, and no other *witness* or *request* events. This trace trivially satisfies strong sender invariance (as $a \notin \mathsf{H\_ID}$) but not weak authentication. This example is a bit contrived, but we give a more realistic example (Protocol 2) in §3.1. □

We now examine the relationship between the two types of sender invariance itself.

Proposition 2 *Strong sender invariance is a strictly stronger security goal than sender invariance.*

Proof. As before, we show that strong sender invariance is at least as strong as sender invariance by showing that any trace satisfying the stronger form also satisfies the weaker one. Consider an arbitrary trace that satisfies strong sender invariance, and consider any event of the form $request(b, id, v, m)$ in the trace, again for arbitrary values b, id, v, and m of the respective types. We have to show that this event is preceded on the trace by the event $witness(b, id, v', m)$ for some $v' \in \mathsf{Vars}$. This holds for $v = v'$, since the trace satisfies strong sender invariance, which requires that $witness(id, b, v, m)$ must precede said *request* event. As we have not assumed anything about the arguments of the *request* event and its position in the trace, this holds for all such *request* events, which shows that sender invariance holds of the trace.

A trivial example to show that sender invariance does not imply strong sender invariance is a trace that contains $request(b, id, v, m)$, preceded by $witness(id, b, v', m)$ for arbitrary constants b, id, v, v', and m, where $v \neq v'$, and such that the trace contains no other *witness* and *request* events. This satisfies sender invariance, but not strong sender invariance. Another example is Protocol 1, which will be discussed in the following section. □

It follows from these propositions that there is a hierarchy of security goals in which weak authentication is strongest, followed by strong sender invariance, and finally sender invariance itself. Specifically, we have seen that strong sender invariance is precisely weak authentication in which pseudonyms are used in place of true agent names. We can observe the same of sender invariance, modulo the fact that the agreement of protocol variables is also ignored. As we will discuss in the next section, this result also illustrates the potential to take existing tools for the automated analysis of authentication protocols and directly use them for the analysis of (strong) sender invariance as well.

3 Analyzing Sender Invariance

We now show how to apply automated tools to analyze (strong) sender invariance protocols. We illustrate this with a case study: the formal analysis of two protocols that use the Purpose-Built Keys Framework.

Classically, the model checking problem $M \vDash \varphi$ verifies whether a model M of a system fulfills a specification of the goal φ. We have analyzed our case-study protocols using the On-the-Fly Model Checker OFMC [5–7], a state-of-the art tool for protocol analysis. OFMC is one of the back-ends of the AVISPA Tool, in which protocols are specified using the *High-Level Protocol Specification Language HLPSL* [3, 11].[3] Protocol models built using this specification language capture the requirements for formalizing sender invariance that we identified in Fig. 1. OFMC allows the modeler to specify φ, where goals are specified negatively as attack states. Thus, for the analyses described in the coming sections, we were able to translate the formulas STRONGSI and SI into HLPSL directly.

Note that while OFMC allows user-defined goals, some model checkers for security protocols consider a fixed, built-in set of goals φ tailored to the application domain: in general, authentication and secrecy. In the previous section, however, we showed that both forms of sender invariance can be seen as a generalization of weak authentication. Based on this, we can identify the following additional requirements on protocol models for use with such fixed-goal model checkers. If

– one can construct protocol specifications in which authentication is performed on pseudonyms,
– honest pseudonyms can be distinguished in the model from those belonging to the intruder, and
– in the case of SI, agreement on protocol variables can be ignored,

then model checkers that are tailored to check WAUTH can be employed, out of the box, to also check STRONGSI and SI.

[3] The HLPSL is an expressive, modular, role-based, formal language that allows for the specification of control flow patterns, data structures, complex security properties, as well as different cryptographic operators and their algebraic properties. The AVISPA Tool automatically translates a user-defined security problem into an equivalent specification written in the rewrite-based formalism *IF* (for *Intermediate Format*). An IF specification describes an infinite-state transition system amenable to formal analysis: this specification is input to OFMC and the other back-ends of the AVISPA Tool, which implement a variety of techniques to search the corresponding infinite-state transition system for states that represent attacks on the intended properties of the protocol.

3.1 Case Study: Purpose-Built Keys

Analyzing Protocol 1 We return to Protocol 1, introduced in §2.1. We constructed a formal model of the Protocol 1 in the HLPSL specification language and analyzed it in a scenario with a bounded number of protocol sessions. For brevity, we omit the HLPSL specification itself and describe only the aspects most important for the analysis.

In our model of the protocol, honest agents generate new PBKs freshly for each session, and these are added to the set $\mathsf{H\_ID}$ upon generation. HLPSL supports the modeling of sets, and using this we maintain a single, global $\mathsf{H\_ID}$. The intruder, who is active and has full Dolev-Yao [13] control over the network as described, may also generate fresh PBKs (and may apply the function H to generate valid pseudonyms), but he may not add them to $\mathsf{H\_ID}$. He may, however, replay any of the keys in that set. We assume that the responder wants sender invariance guarantees on the contents of every message after the initialization phase. Thus, the set $\mathsf{Vars} = \{Msg, N_B\}$, and in the model the responder issues two *request* facts after receiving message 4. In turn, the initiator role issues *witness* facts upon sending messages 2 and 4.

OFMC employs a number of symbolic techniques to perform falsification (by finding an attack on the input protocol) and bounded verification, i.e. verification for a finite number of protocol sessions. In our analysis scenario, we assumed four concurrent protocol sessions (four instances each of the initiator and responder roles). In OFMC, these sessions are specified symbolically, so we need not specify concretely which agent plays which role. Rather, the identities of the agents participating in each session are given simply as variables, and OFMC searches symbolically through all possible assignments of these variables. Our first analysis used SI, sender invariance, as the goal of the protocol. As our analysis found no attacks, this amounts to bounded verification of all possible analysis scenarios consisting of four protocol sessions. This shows that PBK is indeed a strong mechanism for providing sender invariance.

Strong Sender Invariance We also analyzed Protocol 1 against the goal of strong sender invariance. Recall that, by design, sender invariance ignores the interpretation that agents assign to messages, which we express via the protocol variables in set Vars. Thus, protocols guaranteeing sender invariance may well suffer from vulnerabilities in which an intruder succeeds in causing a confusion between the interpretation assigned to a message by the sender and that assigned by the receiver. Indeed, our analysis found an attack on strong sender invariance, shown in Fig. 2.

In this execution, the intruder intercepts the purpose-built key PBK_A and wishes to pass himself off to B as someone who possesses the associated private key. To this end, after receiving A's second message, he replies with the challenge Msg_I, the payload message he actually wants to send to B. A replies in good faith, signing the challenge with ${PBK_A}^{-1}$. In a second session, i then claims PBK_A as his own purpose-built key and sends the signed payload message

1. $A \rightarrow i : PBK_A$
2. $A \rightarrow i : \{Msg_A\}_{PBK_A{}^{-1}}.H(PBK_A)$
3. $i \rightarrow A : Msg_I.H(PBK_A)$
4. $A \rightarrow i : \{Msg_I\}_{PBK_A{}^{-1}}.H(PBK_A)$

$1'.\ i \rightarrow B : PBK_A$

$2'.\ i \rightarrow B : \{Msg_I\}_{PBK_A{}^{-1}}.H(PBK_A)$

Fig. 2. An attack on Protocol 1 that violates STRONGSI

1. $A \rightarrow B : PBK_A$

....

2. $A \rightarrow B : \{tag_1.Msg\}_{PBK_A{}^{-1}}.H(PBK_A)$
3. $B \rightarrow A : N_B.H(PBK_A)$
4. $A \rightarrow B : \{tag_2.N_B\}_{PBK_A{}^{-1}}.H(PBK_A)$

Protocol 2: A refined PBK-Based Protocol

$\{Msg_I\}_{PBK_A{}^{-1}}$. Recall that we assume that messages 2 through 4 may be repeated multiple times to transmit payload data over the lifetime of a pseudonym, therefore the intruder can even perform the challenge-response exchange with B as soon as A sends another payload message.

This attack represents a confusion of the protocol variables assigned to message 4 by A and message $2'$ by B. Although A did indeed once send the message $\{Msg_I\}_{PBK_A{}^{-1}}.H(PBK_A)$, she sent it interpreting it as message 4 of the protocol and thus assigned $N_B = Msg_I$, whereas B interprets it as message 2 upon receipt, assigning $Msg = Msg_I$. Thus, this attack violates the goal of strong sender invariance, but not sender invariance itself. As discussed in §2.5, Protocol 1 illustrates that SI is strictly weaker than STRONGSI.

Analyzing Protocol 2 Protocol 2 shows an alternative example of a protocol that uses the PBK framework. It is identical to Protocol 1 save for the fact that so-called tags have been added to messages 2 and 4. The tags tag_1 and tag_2 are intended as identifiers for the signed messages that signify the purpose of the signature. They avoid, for instance, that B or an intruder can bring A to sign arbitrary data without indicating what the signature is intended for.

We used OFMC to analyze Protocol 2 in the same setting as used in our analysis of Protocol 1. The *witness* and *request* facts generated contain protocol variables indicating the interpretation assigned by the agents, $m2$ and $m4$ for messages 2 and 4, respectively. As specified in the formula STRONGSI, matching *request* and *witness* events must agree on this interpretation. Our analysis results show that, for scenarios consisting of four protocol sessions, Protocol 2 is safe from attacks on strong sender invariance. From Proposition 2, we can conclude that Protocol 2 is thus safe from attacks on sender invariance as well.

1. $A \to B : PBK_A$
....
2. $A \to B : \{tag_1.\{K\}_{PK_B}\}_{PBK_A^{-1}}.H(PBK_A)$
3. $B \to A : \{N_B.B\}_K.H(PBK_A)$
4. $A \to B : \{N_B.H(PBK_A)\}_K.H(PBK_A)$

Protocol 3: A PBK-Based Protocol in which A knows B's public key in advance

4 Varying Amounts of Pre-Shared Information

Sender invariance appears to be an appealing security goal that is appropriate for settings such as those that arise in mobile networks where users do not know one another in advance. Naturally, situations arise that fall between the case in which two agents involved in a protocol run initially share cryptographically authenticated information and the other extreme case in which they share nothing. Perhaps the most prevalent example of this arises in E-commerce situations in which the selling party presents a public-key certificate signed by a well-known certification authority, while the buyer's credentials comprise, at most, a username and password set up via an informal registration procedure.

In cases like these, where the amount of information shared between protocol participants is greater, we can achieve accordingly stronger security goals. We illustrate this with a brief example: Protocol 3 is another protocol that employs the PBK framework. Unlike in the previous ones, however, we assume that the initiator A knows the public key PK_B of the responder B in advance. After A sends her PBK, she generates a new session key K for use between A and B. She encrypts this key with PK_B and signs it, together with a tag indicating the purpose of the signature. B responds, encrypting a nonce N_B together with his name using the new key K. A responds to the challenge and returns the nonce N_B together with her pseudonym $H(PBK_A)$ (twice, once encrypted and once in plaintext).

In discussing the previous protocols, we focused on role B's guarantee of (strong) sender invariance with respect to role A. The agent playing role A, however, could say little or nothing about the security of her communication with B. As in Protocol 2, B is ensured sender invariance with respect to role A. Here, however, A is able to leverage the fact that she knows B's public key to send a new session key K secretly. Messages 3 and 4 serve to ensure recentness and key confirmation to both parties. Subsequent communication secured with the key K should then enjoy the following security properties:

- secrecy of the communication,
- responder B should be guaranteed sender invariance with respect to role A, and
- initiator A should be guaranteed authenticity of the communication from B (as only B should have been able to decrypt K).

This simple example shows how a pre-existing relationship, even a unilateral one, enables significantly greater security. A more prominent example of this is found in the use of SSL/TLS [12] in E-commerce. Most E-commerce applications employ server certificates for servers to authenticate themselves to clients, but forgo the use of client certificates. Hence, this situation is analogous to the one just described: the client is guaranteed the authenticity of the server, but—at least on the transport layer—the server can only refer to the client via a pseudonym.

Overall, as mobile and ad-hoc networks gain ground, we expect to see an increase in situations in which some measure of information, though perhaps not as much as is assumed by traditional authentication protocols, is initially shared. It is therefore important to precisely understand what security goals are achievable in the different settings.

5 Conclusion

Sender invariance is a variant of authentication, with the difference that the identity of a sender is not known to the receiver, but rather the sender is identified by a pseudonym. The key point is that sender invariance can be achieved out of nothing, i.e. even when the agents have no previous security relationship (like shared keys, public keys, or a relationship via a trusted third party) and therefore classical authentication cannot be achieved.

In this paper, we have formalized two forms of sender invariance as variants of classical authentication, and showed that these goals form a hierarchy in the sense that one goal is strictly stronger than the other, with classical authentication being the strongest.

This relationship with classical authentication has allowed us to formalize sender invariance goals for an existing protocol analysis system, the OFMC back-end of the AVISPA Tool. As a case study, we have analyzed protocols using the Purpose-Built Keys Framework (PBK [10]), showing that a naïve protocol implementation has vulnerabilities but still provides a weak form of sender invariance, while an improved implementation with tags provides strong sender invariance.

As future work, we plan to consider further sender invariance protocols like the secure neighbor discovery and binding update protocols of Mobile IPv6 [2, 15]. Moreover, we plan to examine a further generalization of the view on sender invariance and, conversely, investigate "receiver invariance", i.e. the property that only the party that sent the first message (and who created the respective pseudonym) can read the messages directed to him. Receiver invariance can then be the counterpart of classical secrecy goals in the realm of sender invariance protocols.

References

1. M. Abadi. Private authentication. In *Proceedings of PET'02*, LNCS 2482, pages 27–40. Springer, 2002.

2. J. Arkko, J. Kempf, B. Zill, and P. Nikander. RFC3971 – SEcure Neighbor Discovery (SEND). March 2005.
3. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. Heàm, J. Mantovani, S. Moedersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Proceedings of CAV'05*, LNCS 3576, pages 281–285. Springer, 2005. The AVISPA Tool is available at `http://www.avispa-project.org`.
4. T. Aura. Cryptographically generated addresses (CGA). RFC 3972, IETF, Mar. 2005.
5. D. Basin, S. Mödersheim, and L. Viganò. Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols. In *Proceedings of CCS'03*, pages 335–344. ACM Press, 2003.
6. D. Basin, S. Mödersheim, and L. Viganò. Algebraic intruder deductions. In *Proceedings of LPAR'05*, LNAI 3835, pages 549–564. Springer, 2005.
7. D. Basin, S. Mödersheim, and L. Viganò. OFMC: A Symbolic Model-Checker for Security Protocols. *International Journal of Information Security*, 4(3):181–208, 2005.
8. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of CRYPTO'93*, LNCS 773, pages 232–249. Springer, 1994.
9. C. Boyd. Security architectures using formal methods. *IEEE Journal on Selected Areas in Communications*, 11(5):694–701, 1993.
10. S. Bradner, A. Mankin, and J. I. Schiller. A framework for purpose built keys (PBK), June 2003. Work in Progress (Internet Draft).
11. Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Proceedings of SAPS'04*, pages 193–205. Austrian Computer Society, 2004.
12. T. Dierks and C. Allen. RFC2246 – the TLS protocol version 1, Jan. 1999.
13. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
14. D. Gollmann. What do we mean by Entity Authentication? In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 46–54. IEEE Computer Society Press, 1996.
15. D. Johnson, C. Perkins, and J. Arkko. RFC3775 – Mobility Support in IPv6. June 2004.
16. G. Lowe. A hierarchy of authentication specifications. In *Proceedings of CSFW'97*, pages 31–43. IEEE Computer Society Press, 1997.
17. P. Nikander, J. Kempf, and E. Nordmark. RFC3756 – IPv6 Neighbor Discovery (ND) Trust Models and Threats. May 2004.