Compositional Verification of Secure Streamed Data: a Case Study with EMSS *

Fabio Martinelli, Marinella Petrocchi, and Anna Vaccarelli

Istituto di Informatica e Telematica - C.N.R.
Area della Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, Italy.
e-mail: {fabio.martinelli, marinella.petrocchi, anna.vaccarelli}@iit.cnr.it

Abstract. We consider an instance of the EMSS protocol proposed in [19], authenticating streamed data in the presence of packet loss. We formally prove the integrity property of the instance by applying a compositional proof rule that allows us to check a specification with an arbitrary number of parallel processes. We argue that our approach may be applied to a wider class of stream signature protocols.
Key words: Security, Compositional Analysis and Verification, Digital Streams, Integrity.

1 Introduction

Increasing trend to distribute streamed radio and video over the Internet must provide sufficient security guarantees. In particular, so called stream signature protocols were born with the intent to efficiently solve the problem to sign digital streams, i.e. long, possibly infinite, sequence of bits. This class of protocols, designed for open architectures, makes use of hashing techniques and thrifty use of standard digital signatures to ensure the integrity of the stream.

Roughly speaking, an intrinsic infinite nature marks the class of stream signature protocols from the standard cryptographic ones, in the sense that a sender broadcasts a continuous (and possibly unbounded) stream of messages to an arbitrary number of receivers. Further, receivers use information retrieved in earlier packets to legitimate later packets or vice-versa. In wireless (or semi-wireless) environments, the mobile receiver may leave a cell without prior negotiation and reenter any other cell an arbitrary number of times.

Given these peculiarities, the use of formal techniques to analyze stream signature protocols have recently raised an increasing interest among researchers. In particular, in [3] it is shown how to build a finite model of the TESLA protocol [18], despite the possibly unbounded stream of messages (and cryptographic keys) broadcasted by a sender. This allows the authors of [3] to exploit model checking techniques for the formal verification of TESLA. The work in [1] has previously analyzed the same protocol making use of the theorem prover TAME.

* Work partially supported by MURST Progetto "Metodi Formali per la Sicurezza ed il Tempo" (MEFISTO); by MIUR project COVER; by Microsoft Research (Cambridge) and by a CSP grant for the project "SeTAPS II".

Our approach is quite different and focuses its attention on the verifiability of a system with an arbitrary number of components. In this paper we show how to apply a compositional principle allowing us to compose safely processes, preserving the security properties they enjoy. Broadly speaking, a compositional principle gives sufficient conditions to conclude that the parallel composition of two (or more) processes satisfies a certain property, provided that the single processes by themselves satisfy the same property. Compositional reasoning is often useful. An interesting application field is indeed the analysis of systems with an arbitrary number of components (like the possibly arbitrary number of receivers participating through a stream signature protocol).

The principle we are going to exploit in this paper was first discussed in [11] for the GNDC schema of properties, defined in [4, 5]. In turn, the GNDC schema is based on the notion of non-interference, [7]. We aim at applying the principle to stream signature protocols for the verification of the integrity property, where integrity means, informally, that the information accepted by a receiver is exactly what the sender intended.

Digital streams are usually sent over UDP, the User Datagram Protocol, [20]. UDP is considered to be an unreliable transport protocol. When UDP sends packets over a network, it just sends them and forgets about them. This does not mean that UDP is ineffective, only that it does not handle reliability. If a stream is received incomplete, we would still like to be able to prove the integrity of all the packets that were not lost. Stream signature protocols dealing with the problem of packet loss have been recently proposed, [8, 16, 17, 19]. Here, the main target of our analysis will be the EMSS protocol, [19].

The main contribution of this paper is the formal capability to check a stream signature protocol with an arbitrary number of receivers, contrary to previous work in the area, [1, 3] (the target of our analysis however being different from the TESLA protocol considered in those papers). To present our methodology, an instance of the EMSS protocol has been formally analyzed with compositional proof rules and the results are reported in the paper. Earlier results in [12] dealt with the protocol in [6], not designed for dealing with packet loss. Finally, this paper is an extended and revised version of [15].

The paper is organized as follows. In Section 2, we present the formal language we use for the description of cryptographic protocols. In Section 3, we describe the EMSS stream signature protocol in more detail. Section 4 recalls a general schema for defining security properties and illustrates a compositional result to establish if a system enjoys security properties defined by means of the general schemes. Sections 5 shows how to apply the compositional results to successfully prove the correctness of an instance of EMSS in terms of packets' integrity. In Section 6 we report some concluding remarks and discuss about related and future work.

2 A formal language for the description of protocols: Crypto-CCS

A language, a slight modification of CCS process algebra, is adopted for the description of cryptographic protocols. It makes use of cryptographic-modeling constructs and deals with confidential values, hence the name Crypto-CCS [4, 5, 14]. The Crypto-CCS model consists of a set of sequential agents able to communicate by exchanging messages.

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

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

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

Inference systems model the possible operations on messages. These systems consist of a set of rules r:

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

where $m_1, \ldots, m_n$ is a set of premises (possibly empty) and m_0 is the conclusion. An instance of the application of the rule r to closed messages m_i is denoted as $m_1 \quad \ldots \quad m_n \vdash_r m_0$. Given an inference system, we can define a *deduction function* $\mathcal{D}$ s.t. if ϕ is a finite set of closed messages, then $\mathcal{D}(\phi)$ is the set of closed messages that can be deduced starting from ϕ by applying instances of the rules in the system.

The control part of our language consists of compound systems, basically sequential agents running in parallel. The terms of our language are generated by the following grammar:

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

where $m, m', m_1, \ldots, m_n$ are *closed* messages or variables, $x, x_1, \ldots, x_n$ are message variables, $c \in Ch$, a finite set of channels, ϕ is a finite set of *closed* messages, C is a subset of Ch. Constants are defined as follows: $E(x_1, \ldots, x_n) \doteq A$ where A is a Crypto-CCS agent which may contain no free variables except $x_1, \ldots, x_n$ which must be distinct. The informal semantics of sequential agents and compound systems is as follows:

- $\mathbf{0}$ is the process that does nothing.
- $p.A$ is the process that can perform an action according to the particular prefix construct p and then behaves as A:

  - $c!m$ denotes a message m sent on channel c;
  - $c?x$ denotes the receiving of a message m on channel c. The received message replaces the variable x.
- $A + A_1$ represents the non deterministic choice between A and A_1.
- $[m_1 \dots m_n \vdash_r x]A; A_1$ is the inference construct. If, by applying an instance of rule r with premises $m_1 \dots m_n$, a message m can be inferred then the process behaves as A (where m replaces x), otherwise it behaves as A_1. This is the message-manipulating construct of the language. For instance,

$$[m \quad sk(y) \vdash_{sign} x]A; \mathbf{0}$$

  is the process that uses the rule $sign$ to obtain a digitally signed message from plaintext m and private key $sk(y)$ and then behaves like A, or it gets stuck.
- $[m = m']A; A_1$ is the match construct to check message equality: if $m = m'$ then the system behaves as A, otherwise it behaves as A_1.
- $E(m_1, \dots, m_n)$ is a constant process that behaves like the respective defining term P (see Tab. 1) where all the variables $x_1 \dots x_n$ are substituted with messages $m_1 \dots m_n$.
- A compound system $S_1 \mid S_2$ denotes the parallel execution of S_1 and S_2. $S_1 \mid S_2$ performs an action p if one of its sub-components performs p. A synchronization or internal action, denoted by the symbol τ, may take place whenever S_1 and S_2 are able to perform two complementary actions, i.e. send-receive actions on the same channel.
- A compound system $S \setminus C$ allows only visible actions whose channels are not in C. (Internal action τ being the invisible action).
- The term A_ϕ is a single sequential agent whose knowledge, i.e. the set of messages which occur in its term, is described by ϕ. The agent's knowledge increases either when it receives messages (see rule (?) in Tab. 1) or infers new messages from the messages it knows (see rule $\mathcal{D}$ in Tab. 1). For every sequential agent A_ϕ, we require that all the closed messages that appear in A belong to its knowledge ϕ.

2.1 Operational semantics and auxiliary notions.

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

The operational semantics of a Crypto-CCS term is described by means of the *labeled transition system* (*lts*, for short) $\langle \mathcal{P}, Act, \{\xrightarrow{a}\}_{a \in Act} \rangle$, where $\{\xrightarrow{a}\}_{a \in Act}$ is the least relation between $tCryptoSPA$ processes induced by the axioms and inference rules of Tab. 1

The expression $S \xrightarrow{a} S'$ means that the system can move from the state S to the state S' through the action a. The expression $S \Longrightarrow S'$ denotes that S

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

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

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

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

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

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

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

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

Table 1. Operational semantics, where the symmetric rules for $|_1, |_2, \backslash_1, +_2$ are omitted.

and S' belong to the reflexive and transitive closure of $\xrightarrow{\tau}$; let $\gamma = \alpha_1 \ldots \alpha_n \in (Act \backslash \{\tau\})^*$ be a sequence of actions. Then, $S \stackrel{\gamma}{\Longrightarrow} S'$ if and only if there exist $S_1, S_2, \ldots, S_{n-1} \in \mathcal{P}$ such that $S \stackrel{\alpha_1}{\Longrightarrow} S_1 \stackrel{\alpha_2}{\Longrightarrow} S_2 \ldots \stackrel{\alpha_{n-1}}{\Longrightarrow} S_{n-1} \stackrel{\alpha_n}{\Longrightarrow} S'$.

As behavioral relations among Crypto-CCS terms, in the following we will be mainly interested in trace inclusion (equivalence) and (weak) simulation.

Definition 1. *We say that the traces of P are included in the traces of Q ($P \leq_{trace} Q$) whenever, if $P \stackrel{\gamma}{\Longrightarrow} P_1$ then $Q \stackrel{\gamma}{\Longrightarrow} Q_1$. We write that $P =_{trace} Q$ iff $P \leq_{trace} Q$ and $Q \leq_{trace} P$.*

Definition 2. *We say that a relation $\mathcal{R}$ among processes is a weak simulation, if for every $(P, Q) \in \mathcal{R}$ we have:*

- *If $P \xrightarrow{a} P', a \neq \tau$, then there exists Q' s.t. $Q \stackrel{a}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$.*
- *If $P \xrightarrow{\tau} P'$ then there exists Q' s.t. $Q \Longrightarrow Q'$ and $(P', Q') \in \mathcal{R}$.*

The union of all weak simulations is a weak simulation and it is denoted by $\prec$. As usual, it holds that if $P \prec Q$ then $P \leq_{trace} Q$.

3 The EMSS Protocol

In [19], Perrig *et al.* presented the Efficient Multi-chained Stream Signature (EMSS) protocol to sign digital streams. EMSS exploits a combination of hash functions and digital signatures and–contrary to previous proposals [6]–achieves (some) robustness against packet loss.

We assume that a sender S wants to send a stream of messages $m_0, m_1, \ldots,$ m_{last} to a set of receivers $\{R_n \mid n \geq 1\}$. The protocol then requires S to divide the stream into packets and send them to the receivers. The basic idea of the construction is the following: a hash of packet P_{i-1} is appended to packet P_i, whose hash is in turn appended to packet P_{i+1} and so on. A signature packet, containing the hash of the final data packet along with a signature, is sent at the end of the stream. To achieve robustness against packet loss (the event of one or more packets loss would break the chain) each packet contains multiple hashes of previous packets and the signature packet signs hashes of multiple packets. [19] uses both deterministic and random distribution of hashes per packet.

Here we focus on a specific instance of the EMSS, viz. the deterministic (1,2) schema, where packet P_i contains hashes of packets $i-1, i-2$ and whose hash is contained in packets $i+1, i+2$. After an initial phase, each packet P_i contains a meaningful payload m_i[1] together with the hashes $h(P_{i-1})$ and $h(P_{i-2})$ of the previous two packets sent. Packets are sent over channels $c_i, 0 \leq i \leq last$. The end of a stream is indicated by a signature packet P_{sign} over channel c_{sign}, containing the hashes of the final two packets, along with a digital signature. The protocol can formally be described as follows.

$$\begin{array}{ll} \text{Packet } P_0 & c_0\ S \rightarrow \{R_n\} : m_0, null, null \\ \text{Packet } P_1 & c_1\ S \rightarrow \{R_n\} : m_1, h(P_0), null \\ \text{Packet } P_i & c_i\ S \rightarrow \{R_n\} : m_i, h(P_{i-1}), h(P_{i-2})\ 2 \leq i \leq last \end{array}$$

Let P_{last} be the last packet of the stream. Upon sending P_{last} a signature packet is sent:

$$\text{Sign-Pack} \quad P_{sign} \quad c_{sign}\ S \rightarrow \{R_n\} : \{h(P_{last}), h(P_{last-1})\}_{sk(S)}$$

A packet P_i is said to be verifiable if there exists a path (in terms of hash chains) from P_i to the signature packet. Given a set of verifiable packets, we intend to prove the correctness of the construction in terms of packet integrity, i.e. to assure a receiver that the information he received is exactly what the sender has originally intended.

3.1 Crypto-CCS specifications of the (1,2) EMSS

We present the Crypto-CCS specifications of the (1,2) scheme of the EMSS protocol. The sender and receiver processes can perform sendings and receptions

[1] We assume the sender's private key $sk(S)$ cannot be deduced from the set of messages $\{m_i\}$.

according to the protocol original specifications. Compared with a standard notation like the one in Section 3, the Crypto-CCS representation is more expressive: checks on the received packets are explicitly represented. We remind that the whole formalization, in particular the way a receiver process acts, is based on implementative choices of the authors since some details are not explicitly given in [19].

The sender process is parameterized by variables containing the hashes he should insert in the following packet. With notation x_m we mean "variable x should contain message m".

$S_0(0,0) \doteq$	
$[m_0 \vdash_{tuple} x_{P_0}]$	*Create tuple*
$[x_{P_0} \vdash_{hash} x_{h(P_0)}]$	*Compute hash of* P_0
$(S_1(x_{h(P_0)}, 0) \mid MB_0(x_{P_0}))$	*Output* P_0 *and go to next state*
$S_1(x_{h(P_0)}, 0) \doteq$	
$[m_1 \quad x_{h(P_0)} \vdash_{tuple} x_{P_1}]$	*Create tuple*
$[x_{P_1} \vdash_{hash} x_{h(P_1)}]$	*Compute hash of* P_1
$(S_2(x_{h(P_1)}, x_{h(P_0)}) \mid MB_1(x_{P_1}))$	*Output* P_1 *and go to next state*
$S_i(x_{h(P_{i-1})}, x_{h(P_{i-2})}) \doteq$	
$[m_i \quad x_{h(P_{i-1})} \quad x_{h(P_{i-2})} \vdash_{tuple} x_{P_i}]$	*Create tuple*
$[x_{P_i} \vdash_{hash} x_{h(P_i)}]$	*Compute hash of current packet*
$(S_{i+1}(x_{h(P_i)}, x_{h(P_{i-1})}) \mid MB_i(x_{P_i}))$	*Output* P_i *and go to next state*
$S_{sign}(x_{h(P_{last})}, x_{h(P_{last-1})}) \doteq$	
$[x_{h(P_{last})} \quad x_{h(P_{last-1})} \vdash_{tuple} x_t]$	*Create tuple of final hashes*
$[x_t \quad sk(S) \vdash_{sign} x_{P_{sign}}]$	*Sign the tuple*
$MB_{sign}(x_{P_{sign}})$	*Output the signature packet*

The special process MB is responsible for potentially sending each packet an unbounded number of times, in order to simulate a one-to-many (one-to-all) sending typical of a multicast/broadcast session. The process is parameterized by the packet the sender is to multicast (or broadcast).

$$MB_i(x_{P_i}) \doteq c_i!x_{P_i}.MB_i(x_{P_i}) \quad 0 \leq i \leq last$$
$$MB_{sign}(x_{P_{sign}}) \doteq c_{sign}!x_{P_{sign}}.MB_{sign}(x_{P_i})$$

Among the set of receivers, each process behaves in the same way. The generic receiver process at step i is parameterized by: 1) the two last packets he received (let them be P_{j_1}, P_{j_2}) - over an ideal channel, without packet loss, we have that $P_{j_1} = P_{i-1}$ and $P_{j_2} = P_{i-2}$; 2) a tuple $tup^{i-1}_{\{m_j\}}$. $tup_{\{m_j\}}$ consists of the ordered sequence of payloads among $\{m_j\}_{j=0,1,...last}$ whose corresponding packets' hashes $h(P_j)$ the receiver was able to check[2]. $tup^{i-1}_{\{m_j\}}$ is the tuple updated at step

[2] For the sake of readability we assume the receiver may infer the sequence number of a packet by simply observing the packet itself. Otherwise, we should arrange the receiver with more parameters or arrange a "sequence number" field in the packet structure and let the receiver retrieve it. This could introduce a too clumsy notation.

i, by inserting either $x_{m_{i-2}}$ or $x_{m_{i-3}}$. $tup^{i-1}_{\{m_j\}}$ could be either $(x_{m_{i-2}}, tup^{i-1}_{\{m_j\}})$ or $(x_{m_{i-3}}, tup^{i-1}_{\{m_j\}})$ or it may remain unchanged. Similarly, $tup_{\{m_j\}}$ may either be $(x_{m_{last}}, tup^{last}_{\{m_j\}})$ or $(x_{m_{last-1}}, tup^{last}_{\{m_j\}})$ or $tup^{last}_{\{m_j\}}$.

We model the unreliability of the transmission over UDP by considering that process *Rec* non deterministically chooses whether to receive a packet or not. Finally, we assume that the signature packet P_{sign} is always received (this is likely since in the original protocol multiple copies of the signature packets are sent).

$Rec_0(0,0,0) \doteq$
$Rec_1(0,0,0) +$ — *Packet loss : go to next state, otherwise*
$(c_0?x_{P_0}.$ — *Receive initial packet*
$Rec_1(x_{P_0},0,0))$ — *Go to next state*

$Rec_1(0,0,0) \doteq$
$Rec_2(0,0,0) +$ — *Packet loss : go to next state, otherwise*
$(c_1?x_{P_1}.$ — *Receive packet P_1*
$Rec_2(x_{P_1},0,0))$ — *Go to next state*

$Rec_1(x_{P_0},0,0) \doteq$
$Rec_2(0,x_{P_0},0)+$ — *Packet loss : go to next state, otherwise*
$(c_1?x_{P_1}.$ — *Receive packet P_1*
$[x_{P_1} \vdash_{2-nd} x_{h(P_0)}]$ — *Extract hash of previous packet P_0*
$[x_{P_0} \vdash_{hash} x_{h_{MY}(P_0)}]$ — *Compute my hash $h_{MY}(P_0)$*
$[x_{h(P_0)} = x_{h_{MY}(P_0)}]$ — *Compare the hashes*
$([x_{P_0} \vdash_{1-st} x_{m_0}]$ — *IF equal : extract previous payload*
$Rec_2(x_{P_1}, x_{P_0}, x_{m_0})$ — *Update parameters and go to next state*
$); \mathbf{0}$ — *ELSE abort*
$)$

$Rec_i(x_{P_{j_1}}, x_{P_{j_2}}, tup^{i-1}_{\{m_j\}}) \doteq$
$Rec_{i+1}(x_{P_{j_1}}, x_{P_{j_2}}, tup^{i-1}_{\{m_j\}}) +$ — *Packet loss : go to next state, otherwise*
$(c_i?x_{P_i}.$ — *Receive packet P_i*
$([j_1 = i-1]$ — *Was P_{i-1} received?*
$Rec'_i(x_{P_i}, x_{P_{i-1}}, tup^{i-1}_{\{m_j\}});$ — *Go to Rec'_i; otherwise*
$([j_2 = i-2]$ — *Was P_{i-2} received?*
$Rec''_i(x_{P_i}, x_{P_{i-2}}, tup^{i-1}_{\{m_j\}})$ — *Go to Rec''_i; otherwise*
$)$
$); Rec_{i+1}(x_{P_i}, x_{P_{j_1}}, tup^{i-1}_{\{m_j\}})$ — *Go to next state :*
$)$ — *P_{1-1} and P_{i-2} were not received*
$Rec'_i(x_{P_i}, x_{P_{i-1}}, tup^{i-1}_{\{m_j\}}) \doteq$
$[x_{P_i} \vdash_{2-nd} x_{h(P_{i-1})}]$ — *Extract $h(P_{i-1})$ from P_i*
$[x_{P_{i-1}} \vdash_{hash} x_{h_{MY}(P_{i-1})}]$ — *Compute my hash $h_{MY}(P_{i-1})$*
$[x_{h_{MY}(P_{i-1})} = x_{h(P_{i-1})}]$ — *Compare the hashes*
$([x_{P_{i-1}} \vdash_{1-st} x_{m_{i-1}}]$ — *IF equal : extract m_{i-1} from P_{i-1}*
$Rec_{i+1}(x_{P_i}, x_{P_{j_1}}, (x_{m_{i-1}}, tup^{i-1}_{\{m_j\}}))$ — *Update parameters and go to next state*
$); \mathbf{0}$ — *ELSE : abort*

$Rec''_i(x_{P_i}, x_{P_{i-2}}, tup^{i-1}_{\{m_j\}}) \doteq$

$[x_{P_i} \vdash_{3-rd} x_{h(P_{i-2})}]$	*Extract* $h(P_{i-2})$ *from* P_i
$[x_{P_{i-2}} \vdash_{hash} x_{h_{MY}(P_{i-2})}]$	*Compute my hash* $h_{MY}(P_{i-2})$
$[x_{h_{MY}(P_{i-2})} = x_{h(P_{i-2})}]$	*Compare the hashes*
$([x_{P_{i-2}} \vdash_{1-st} x_{m_{i-2}}]$	*IF equal : extract* m_{i-2} *from* P_{i-2}
$Rec_{i+1}(x_{P_i}, x_{P_{j_1}}, (x_{m_{i-2}}, tup^{i-1}_{\{m_j\}}))$	*Update parameters and go to next state*
$); \mathbf{0}$	*ELSE : abort*

$Rec_{sign}(x_{P_{j1}}, x_{P_{j2}}, tup^{last}_{\{m_j\}}) \doteq$

$c_{sign}?x_{P_{sign}}.$	*Receive signature packet*
$Rec^*_{sign}(x_{P_{sign}}, x_{P_{j1}}, x_{P_{j2}}, tup^{last}_{\{m_j\}})$	*Go to intermediary state* Rec^*_{sign}

$Rec^*_{sign}(x_{P_{sign}}, x_{P_{j1}}, x_{P_{j2}}, tup^{last}_{\{m_j\}}) \doteq$

$[x_{P_{sign}} \quad pk(S) \vdash_{ver} x_{ver}]$	*Verify the signature*
$[j_1 = last]$	*Was* P_{last} *received?*
$Rec'_{sign}(x_{ver}, x_{P_{last}}, tup^{last}_{\{m_j\}});$	*If so, go to* Rec'_{sign}*; otherwise*
$([j_2 = last - 1]$	*Was* P_{last-1} *received?*
$Rec''_{sign}(x_{ver}, x_{P_{last-1}}, tup^{last}_{\{m_j\}});$	*If so, go to* Rec''_{sign}*; otherwise*
$(c_{app}!tup^{last}_{\{m_j\}}.\mathbf{0})$	P_{last} *and* P_{last-1} *were not received.*
$)$	*Send the stream of verifiable payloads to the application level*

$Rec''_{sign}(x_{ver}, x_{P_{last-1}}, tup^{last}_{\{m_j\}}) \doteq$

$[x_{ver} \vdash_{2-nd} x_{h(P_{last-1})}]$	*Extract* $h(P_{last-1})$ *from* P_{sign}
$[x_{P_{last-1}} \vdash_{hash} x_{h_{MY}(P_{last-1})}]$	*Compute my hash* $h_{MY}(P_{last-1})$
$[x_{h_{MY}(P_{last-1})} = x_{h(P_{last-1})}]$	*Compare the hashes*
$[x_{P_{last-1}} \vdash_{1-st} x_{m_{last-1}}]$	*IF equal : extract* m_{last-1} *from* P_{last-1}
$c_{app}!(x_{m_{last-1}}, tup^{last}_{\{m_j\}}).\mathbf{0};$	*Send the stream of verifiable payloads*
$\mathbf{0}$	*to the application level and stop; ELSE abort*

$Rec'_{sign}(x_{ver}, x_{P_{last}}, tup^{last}_{\{m_j\}}) \doteq$

$[x_{ver} \vdash_{1-st} x_{h(P_{last})}]$	*Extract* $h(P_{last})$ *from* P_{sign}
$[x_{P_{last}} \vdash_{hash} x_{h_{MY}(P_{last})}]$	*Compute my hash* $h_{MY}(P_{last})$
$[x_{h_{MY}(P_{last})} = x_{h(P_{last})}]$	*Compare the hashes*
$[x_{P_{last}} \vdash_{1-st} x_{m_{last}}]$	*IF equal : extract* m_{last} *from* P_{last}
$c_{app}!(x_{m_{last}}, tup^{last}_{\{m_j\}}).\mathbf{0};$	*Send the stream of verifiable payloads*
$\mathbf{0}$	*to the application level and stop; ELSE abort*

In the final state Rec_{sign} (along with intermediary states $Rec^*_{sign}, Rec'_{sign}$, Rec''_{sign}) the receiver aims at verifying the digital signature (we assume he has previously retrieved the public key $pk(S)$ corresponding to the private key of the supposed sender). The correct verification of the signature implies the receiver to have guarantees on the integrity of the verifiable payloads. He can now send the stream to his application level to consume it. In our formalization, this is modeled by a scenario where the receiver sends the content of his parameter tuple (the stream accepted) over channel c_{app}. If the verification of the signature in the final state or the equality tests in the previous states do not succeed the receiver should abort.

4 Compositional analysis within GNDC

In this section we recall the general schema *Generalized Non Deducibility on Compositions* (GNDC) for the definition of security properties given in [4, 5] and a compositional proof rule for such a schema, discussed in [11]. The main idea is the following: a system P satisfies property $GNDC^{\alpha}_{\lhd}$ if the behavior of P, despite the presence of a hostile environment X that can interact with P only through a fixed set of channels C, *appears* to be same (w.r.t. a behavioral relation $\lhd$ of observational equivalence) to the behavior of a modified version $\alpha(P)$ of P that represents the *expected* (correct) behavior of P.

The analysis of cryptographic protocols involves specifying a set of messages known by the adversary at the beginning of the computation. This *static* (initial) knowledge of the hostile environment must be bound to a specific set of messages. This limitation is needed to avoid a too strong hostile environment that would be able to corrupt any secret (as it would know all cryptographic keys, etc.). Given an adversary X, we call $ID(X)$ the set of closed messages that syntactically appear in X. This set, intuitively, contains all the messages that are initially known by X. Let ϕ_X be a set of messages representing the static, initial knowledge that we would like to give to X. We want $ID(X)$ to be consistent with ϕ_X. This can be obtained by requiring that all the messages in $ID(X)$ are *deducible* from ϕ_X by means of the *deduction function* $\mathcal{D}$.

The set $\mathcal{E}_C^{\phi_X}$ of processes that can communicate on a subset of public channels C and have an initial knowledge bound by ϕ_X can be therefore defined as follows:

$$\mathcal{E}_C^{\phi_X} = \{X \in \mathcal{P} \mid sort(X) \subseteq C \text{ and } ID(X) \subseteq \mathcal{D}(\phi_X)\}$$

We consider as hostile processes only the ones belonging to $\mathcal{E}_C^{\phi_X}$.
We define the property $GNDC^{\alpha}_{\lhd}$ as follows:

Definition 3. *A process P is $GNDC^{\alpha}_{\lhd}$* iff $\forall X \in \mathcal{E}_C^{\phi_X} : (P \mid X) \setminus C \lhd \alpha(P)$ *where $\lhd$ is a behavioral relation between processes and α is a function between processes.*

For the analysis of safety properties it is enough to consider the trace inclusion relation $\leq_{trace}$ as behavioral relation among the terms of the algebra. When the $\leq_{trace}$ relation is considered, there exists a sufficient criterion for the static characterization (i.e. not involving the universal predicate $\forall$) of $GNDC^{\alpha}_{\lhd}$ properties (for further details see [4, 5]). Generally, $GNDC_{\leq_{trace}}$ properties are not compositional. To get a compositional rule we need to strengthen our requirements on the behavior of the processes.

Definition 4. *We say that a process P is stable w.r.t. ϕ_X, whenever for every X with $ID(X) \subseteq \phi_X$, $(P \mid X_{\phi_X}) \setminus C \stackrel{\gamma}{\Longrightarrow} (P' \mid X'_{\phi'_X}) \setminus C$ then $\mathcal{D}(\phi_X) = \mathcal{D}(\phi'_X)$.*

Basically, a process P is stable when an enemy with a certain knowledge ϕ_X does not increase significantly ϕ_X during the execution of P.

The following compositional rule holds for the $GNDC^{\alpha}_{\leq_{trace}}$ schema (under the assumption that the involved processes are stable).

Proposition 1. *Given ϕ_X and a set of public channels C, assume processes $P_r \in GNDC^{\alpha_r(P_r)}_{\leq_{trace}}$ with $1 \leq r \leq n$ and P_r stable w.r.t. ϕ_X. It follows that $(P_1 \mid \ldots \mid P_n)$ is stable w.r.t. ϕ_X and $(P_1 \mid \ldots \mid P_n) \in GNDC^{\alpha_1(P_1) \mid \ldots \mid \alpha_n(P_n)}_{\leq_{trace}}$.*

5 Compositional Analysis of the (1,2) EMSS Protocol

Our goal is to apply the previous compositional rule for checking that the (1,2) EMSS scheme guarantees integrity of the delivered stream even in presence of an adversary. The specifications of the (1-2) EMSS scheme, namely the sender S_0 and the receiver Rec_0, are given in Subsection 3.1. The general system with n receivers may be considered as $S_0 \mid \overbrace{Rec_0 \mid \ldots \mid Rec_0}^{n}$.

We formally define integrity in the GNDC schema as the ability to accept only the message m_i by a receiver as the $i-th$ message sent by the sender (assuming m_i is not lost). Assume that a receiver signals the acceptance of a stream of messages as a legitimate one, by issuing it, as a unique list of messages, on a special channel c_{app}. Thus, let α_{int} be $Spec_{sign} = \sum_{s \in streams} c_{app}!s.\mathbf{0}$, where $streams$ is the set of all the possible ordered sub streams of $m_0 \ldots m_{last}$.

Definition 5. *A system P, consisting of a sender of a stream of messages $\{m_i\}$ and a receiver, enjoys the integrity property whenever $P \in GNDC^{\alpha_{int}}_{\leq_{trace}}$.*

Basically, it means that the receiver accepts exactly a subset of the messages m_i in the correct order even in presence of an adversary. The key point is that the intruder will never acquire the private key of the sender to successfully sign the final packet of the stream. Note that in a multi-receiver environment with one sender, a protocol guarantees integrity whenever each receiver accepts only the stream of messages that the sender wishes to deliver. In our case, the specification for n receivers is simply the parallel composition of α_{int} n-times.

We may prove that S_0, Rec_0 are stable w.r.t. the following initial knowledge ϕ_X:

$$\phi_X = \{P_0\} \cup \{P_1\} \cup \{P_i \mid i = 2, \ldots, last\} \cup \{pk(S), P_{sign}\}$$

Proposition 2. *S_0 and Rec_0 are stable w.r.t. ϕ_X.*

Actually, we include in the initial knowledge ϕ_X the messages an adversary would be able to add to his knowledge by eavesdropping on a run of the protocol. This implies that we arrange for an intruder to have the most powerful means to act since the beginning of the computation. If the protocol satisfies the integrity property in this very hostile environment then it means that it will satisfy this property in a less powerful one (this may be formally justified).

We may prove that S_0 enjoys $GNDC^{\mathbf{0}}_{\leq_{trace}}$ and Rec_0 enjoys $GNDC^{\alpha_{int}}_{\leq_{trace}}$, that is to say for all $X \in \mathcal{E}_C^{\phi_X}$ we have $(S_0 \mid X)\backslash C \leq_{trace} \mathbf{0}$ and $(Rec_0 \mid X)\backslash C \leq_{trace} \alpha_{int}$. This may be done by finding a suitable weak simulation relation between $(S_0 \mid X)\backslash C$ and $\mathbf{0}$, and between $(Rec_0 \mid X)\backslash C$ and $Spec_{sign}$ ($\forall X \in \mathcal{E}_C^{\phi_X}$), respectively. The set C of channels over which an intruder is able to communicate is $C = \{c_{sign}\} \cup \{c_i \mid 0 \leq i \leq last\}$.

The weak simulation relation we consider for dealing with the sender specifications is the following:

$$\begin{aligned}\mathcal{R}_{\mathcal{S}} = (((S_i(...) \mid X)\backslash C, \mathbf{0}) \mid X \in \mathcal{E}_C^{\phi_X}, 0 \leq i \leq last) \\ \cup(((S_{sign}(...) \mid X)\backslash C, \mathbf{0}) \mid X \in \mathcal{E}_C^{\phi_X})\end{aligned}$$

The weak simulation relation we consider for dealing with the receiver specifications is the following:

$$\begin{aligned}\mathcal{R}_{\mathcal{R}} = & (((Rec_0(0,0,0) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}) \\ & \cup(((Rec_1(0,0,0) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}) \\ & \cup(((Rec_1(x_{P_0},0,0) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}) \\ & \cup(((Rec_i(x_{P_{j1}}, x_{P_{j2}}, tup_{\{m_j\}}^{i-1}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}, 2 \leq i \leq last) \\ & \cup(((Rec_i'(x_{P_i}, x_{P_{i-1}}, tup_{\{m_j\}}^{i-1}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}, 2 \leq i \leq last) \\ & \cup(((Rec_i''(x_{P_i}, x_{P_{i-2}}, tup_{\{m_j\}}^{i-1}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}, 2 \leq i \leq last) \\ & \cup(((Rec_{sign}(x_{P_{j1}}, x_{P_{j2}}, tup_{\{m_j\}}^{last}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}) \\ & \cup(((Rec_{sign}^*(x_{P_{sign}}, x_{P_{j1}}, x_{P_{j2}}, tup_{\{m_j\}}^{last}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}) \\ & \cup(((Rec_{sign}'(x_{ver}, x_{P_{last}}, tup_{\{m_j\}}^{last}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X}) \\ & \cup(((Rec_{sign}''(x_{ver}, x_{P_{last-1}}, tup_{\{m_j\}}^{last}) \mid X)\backslash C, Spec_{sign}) \mid X \in \mathcal{E}_C^{\phi_X})\end{aligned}$$

$tup_{\{m_j\}}^{i-1}, tup_{\{m_j\}}^{last}$ are lists of meaningful payloads (also updated). By inspection of the possible cases we may show that $\mathcal{R}_{\mathcal{S}}$ and $\mathcal{R}_{\mathcal{R}}$ are weak simulations. We omitted to explicitly put in $\mathcal{R}_{\mathcal{S}}$ and $\mathcal{R}_{\mathcal{R}}$ the pairs in which the first process performs deduction constructs.

Here we give a sketch of the proof dealing with the receiver specifications.

When the first process performs inference (or match) constructs and it gets stuck because an inference rule does not apply, or simply travels to the next state, it can be weakly simulated by whatever process, in particular $Spec_{sign}$. When Rec_0 performs a receiving action, the process on the left may perform a τ action and it can be weakly simulated by whatever process, in particular $Spec_{sign}$. The interesting case is when the first process outputs a tuple of messages $tup_{\{m_j\}}$ over channel $c_{app} \notin C$. In this case, it must be $\{x_{ver}\}_{sk(S)} = P_{sign}$ and, assuming that digital signatures and hash functions cannot be forged, all the messages in $tup_{\{m_j\}}$ must be replaced with one of all the possible ordered sub streams of $m_0 \ldots m_{last}$. This can be weakly simulated by $Spec_{sign}$ that has been defined as the process sending all the possible ordered sub streams of $m_0 \ldots m_{last}$.

Each resulting pair consisting of the derivatives still belong to $\mathcal{R}_{\mathcal{R}}$.

Proposition 3. *$S_0 \in GNDC_{\leq trace}^{\mathbf{0}}$ and $Rec_0 \in GNDC_{\leq trace}^{\alpha_{int}}$.*

The following proposition follows by Proposition 1, 2, 3.

Proposition 4. *$S_0 \mid Rec_0 \in GNDC_{\leq trace}^{\alpha_{int}}$.*

Compositional reasoning is powerful: to check a system with an arbitrary number of components we do not consider the whole system but only the components by themselves and the result simply follows by Proposition 1 where index r is not fixed *a priori* and $P_1 = S_0$ and $P_r, 2 \leq r \leq n$ is Rec_0.

Proposition 5. *The (1,2) EMSS Protocol enjoys integrity for whatever number of receivers.*

6 Conclusions

The compositional analysis can be successfully applied to formally verify integrity properties in a multicast/broadcast environment. (We remind that integrity means, from our point of view, assurance that multicast data are not modified *en-route*.) In particular, we considered as a case study EMSS [19] in its deterministic (1,2) scheme variant. We modeled such a scheme considering communication channels with packet loss. We are able to formally check the system with an arbitrary number of receivers.

Prominent works related to streams verification are those in [1], a formal analysis based on theorem proving techniques to analyze a well known stream authentication protocol (the TESLA protocol [18]) and in [3], where the authors verify the same protocol with model checking techniques.

Notable examples of compositional proof techniques for reasoning about cryptographic protocols may be found in [2, 13]. In [13] a compositional proof system for an environment-sensitive bisimulation has been developed. One main difference from ours is that we consider a weak notion of observation where the internal actions are not visible. (Actually, the authors of [13] leave as a future work the treatment of such a form of equivalence). In [2] the concept of disjoint encryption has been developed and the authors were able to perform compositional reasoning both for secrecy and authentication properties.

Works in progress are the following: 1) applying our compositional proof rules to erasure codes-based solutions like the ones proposed in [16, 17]; 2) extending the analysis to real-time broadcast environments where time synchronization plays an essential role. Compositionality being a fundamental issue in static analysis approaches, we leave as a future work the comparison of our approach with the one proposed in [9, 10], based on type systems for checking authenticity/integrity properties.

Acknowledgments. We would like to thank the anonymous referees for their helpful comments.

References

[1] M. Archer. Proving Correctness of the Basic TESLA Multicast Stream Authentication Protocol with TAME. In *Proc. of WITS'02*, 2002.

[2] M. Boreale and D. Gorla. On Compositional Reasoning in the Spi-Calculus. In *Proc. of FOSSACS'02*, LNCS 2303, 67-81. 2002.

[3] P. Broadfoot and G. Lowe. Analysing a Stream Authentication Protocol using Model Checking. In *Proc. of ESORICS'02*, LNCS 2502, 146-161. 2002.

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

[5] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proc. of FM'99*, LNCS 1708, 794-813. 1999.

[6] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. *Information and Computation*, 165(1):100–116, 2001.

[7] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of IEEE S&P'82*, pages 11–20, 1982.

[8] P. Golle and N. Modadugu. Authenticating Streamed Data in the Presence of Random Packet Loss. In *Proc. of NDSS'01*, 2001.

[9] A.D. Gordon and A. Jeffrey. Authenticity by Typing for Security Protocols. In *Proc. of IEEE CSFW'01*, 126-144. 2001.

[10] A.D. Gordon and A. Jeffrey. Types and Effects for Asymmetric Cryptographic Protocols. In *Proc. of IEEE CSFW'02*, 77-91. 2002.

[11] R. Gorrieri, E. Locatelli, and F. Martinelli. A Simple Language for Real-time Cryptographic Protocol Analysis. In *Proc. of ESOP'03*, LNCS 2618, 114-128. 2003.

[12] R. Gorrieri, F. Martinelli, M.Petrocchi, and A.Vaccarelli. Compositional Verification of Integrity for Digital Stream Signature Protocols. In *Proc. of IEEE ACSD'03*, 142-149. 2003.

[13] J. Guttman and F.J. Thayer. Protocol Independence through Disjoint Encryption. In *Proc. of IEEE CSFW'00*, 24-34. 2000.

[14] F. Martinelli. Analysis of Security Protocols as Open Systems. *Theoretical Computer Science*, 290(1):1057–1106, 2003.

[15] F. Martinelli, M.Petrocchi, and A.Vaccarelli. Analysing EMSS with Compositional Proof Rules for Non-Interference. In *Proc. of WITS'03*, 52-61. 2003.

[16] A. Pannetrat and R. Molva. Efficient Multicast Packet Authentication. In *Proc. of NDSS'03*, 2003.

[17] J. M. Park, E. K. P. Chong, and H. J. Siegel. Efficient Multicast Packet Authentication using Signature Amortization. In *Proc. of IEEE S&P'02*, pages 227–240, 2002.

[18] A. Perrig, R. Canetti, D. X. Song, and D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Proc. of NDSS'01*. The Internet Society, 2001.

[19] A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proc. of IEEE S&P'00*, pages 56–73, 2000.

[20] J. Postel. The User Datagram Protocol - RFC 768, 1980.