Leakiness is Decidable for Well-Founded Protocols*

Sibylle Fröschle

OFFIS & University of Oldenburg, 26121 Oldenburg, Germany
froeschle@informatik.uni-oldenburg.de

Abstract. A limit to algorithmic verification of security protocols is posed by the fact that checking whether a security property such as secrecy is satisfied is undecidable in general. In this paper we introduce the class of well-founded protocols. It is designed to exclude what seems to be common to all protocols used in undecidability proofs: the protocol syntax ensures that honest information cannot be propagated unboundedly without the intruder manipulating it. We show that the secrecy property of leakiness is decidable for well-founded protocols.

Consider the INSECURITY problem that stands behind classical protocol verification: Given a protocol P and an attack goal G, is there a run of protocol P controlled by the Dolev-Yao intruder that obtains G? This problem is well-known to be undecidable in general [13,17]. One can distinguish between two directions to restrict the problem to analyse the decidability border.

One direction is to restrict the sources of infinity the Dolev-Yao intruder can make use of: an attack may involve messages of unbounded size, an unbounded number of freshly generated data, and an unbounded number of sessions. This direction is well-investigated and many positive results have been obtained. INSECURITY turns out to be NP-complete when the number of sessions is bounded [23,7], and EXPTIME-complete when both the number of freshly generated data and the message size is bounded [12,14]. INSECURITY remains undecidable when only one of these two restrictions is imposed. A recent survey of this area can be found in [14].

The second direction for borderline investigations is to leave the sources of infinity a priori unconstrained, but impose restrictions on the message format. In [10,9] INSECURITY was shown to be PTIME decidable for *ping-pong* protocols. These protocols have a very restricted message format that makes it possible to formalize them by a form of context-free grammars. More directly motivated by protocol verification, works by Lowe [18] and Ramanujam and Suresh [20,22,21] investigate decidability when imposing conditions that make encrypted messages context-explicit. The idea is that such protocols merely satisfy the prudent engineering practice recommended by Abadi and Needham [1]. These works achieve

* This work is partially supported by the *Niedersächsisches Vorab* of the Volkswagen Foundation and the Ministry of Science and Culture of Lower Saxony as part of the *Interdisciplinary Research Center on Critical Systems Engineering for Socio-Technical Systems*.

R. Focardi and A. Myers (Eds.): POST 2015, LNCS 9036, pp. 176–195, 2015.
DOI: 10.1007/978-3-662-46666-7_10

decidability results for the problem of non-secrecy without temporary secrets, which we call LEAKINESS here.

Our Contribution. In this paper, we tackle the decidability of context-explicit protocols 'from the top', trying to pinpoint in an abstract manner why the usual undecidability reductions do not carry over to such protocols. More concretely, we introduce the class of *well-founded protocols*. It is designed to exclude what seems to be common to all protocols used in undecidability results: the message format allows that honest information can be propagated unboundedly without the intruder manipulating it. We prove that LEAKINESS is decidable for well-founded protocols. Our class strictly contains those of [18,20,22,21].

Related Work. In [18] Lowe obtains a small model property (from which decidability of LEAKINESS is immediate) by a condition that requires that encrypted components are textually distinct, and that each encrypted component includes all protocol roles. Together this ensures that every encryption that occurs in a protocol run can be uniquely assigned to a protocol position and the set of agents involved in the run. The *structured* protocols of [20] introduce a condition that is similar to the first part of Lowe's: between any two terms that occur in distinct communications, no encrypted subterm of one can be unified with a subterm of the other. In the full version of this paper [22] the authors obtain NEXPTIME decidability of LEAKINESS for their class of *context-explicit* protocols: these protocols are structured and in addition require that each encryption to be sent out is tagged by a freshly generated nonce. This ensures that each instantiation of an encryption can be traced back to exactly one session. In [21] Ramanujam and Suresh obtain decidability of LEAKINESS for their *tagged* protocols. These are essentially an instance of their context-explicit protocols: the structured condition is implemented by using constants to identify encrypted subterms; they also require the additional dynamic tagging with freshly generated nonces. The novelty of [21] lies in the fact that the result also works for untyped messages, and hence, unbounded message length.

The three results [18,22,21] have in common that they establish a small model property: if there is an attack then there is a small attack and the problem reduces to checking protocol runs with a bounded number of events. The three works do not allow composed keys nor blind forwarding of ciphertexts. Since [18] and [22] work with a typed algebra together with the latter this means that the message size is bounded, and decidability follows from the small model property. [21] lifts the typing restriction, and thereby admits messages of unbounded size. In addition to the small model property it is shown that if there is an attack then there is a well-typed attack.

In [2] Arapinas and Duflot provide a general approach for bounding the size of messages in an attack: they introduce a condition of well-formedness and show that a protocol admits an attack iff it admits a well-typed attack for a particular typing system that bounds message size. They also show that the tagging scheme of Blanchet and Podelski [3] implements well-formedness. The tagging system is light in that it only introduces a different constant in each encryption. The tagging system is used in [3] to enforce termination of a resolution-based

verification method. (The verification method is approximate so this does not give a decidability result.) In [5] the approach of [2] is extended to equivalence checking and a more general typing system. As an application decidability of trace equivalence for tagged protocols is obtained for a fixed number of nonces. The setting admits symmetric encryption and assumes session identifiers.

Dougherty and Guttman are first to apply the idea of context-explicitness in a rich algebraic setting [11]. They introduce a class of lightweight Diffie-Hellman protocols with simple signatures. The simple signatures are defined by requiring an ordering on the occurrences of signatures in a protocol to be acyclic. They obtain a small model property for their class, which together with other algebraic results leads to decidability of their security goals for lightweight Diffie-Hellman protocols.

Synopsis. In Section 1 we present the necessary definitions. In Section 2 we introduce *well-founded protocols* motivated by a notion of *honest causality*, and prove their characteristic property. In Section 3 we introduce a normal form for intruder deductions and protocol runs, so-called *well-structured source trees and bundles*. We obtain two structural insights on well-structured bundles. In Section 4 this allows us to transform *honest cause components* of non-leaky bundles into bundles. The transformation also works for minimal leaky bundles and preserves leaks. Altogether, this means that the size of minimal leaky bundles is bounded by the size of honest cause components. (Indeed, this holds for protocols in general.) For well-founded protocols we obtain a bound on the size of honest cause components, and thereby of minimal leaky bundles. With this decidability is immediate. The ideas behind this work were presented at [15]. A full version of the paper can be found on the web page of the author.

1 Preliminaries

Terms and Messages. Let Atoms be a set of atomic messages or *atoms*, and AVars a set of variables for atoms. Then $\mathit{ATerms} = \mathit{Atoms} \cup \mathit{AVars}$ is the set of atomic terms. The set of atoms, and variables for atoms respectively, can be be further structured into several atomic message types. Here we only assume a set of agent names $\mathit{Agents} \subset \mathit{Atoms}$, and a set of variables for agent names $\mathit{Vars}_{\mathit{Agents}}$ respectively. Set $\mathit{Terms}_{\mathit{Agents}} = \mathit{Agents} \cup \mathit{Vars}_{\mathit{Agents}}$. Moreover, let Vars be a set of variables to present any message.

The set of *terms*, denoted by Terms, is generated from the set of basic terms $\mathit{ATerms} \cup \mathit{Vars}$ by the following operators:

- $\mathit{priv}(ag)$ where $ag \in \mathit{Terms}_{\mathit{Agents}}$,
- $\langle t_1, t_2 \rangle$ where $t_1, t_2 \in \mathit{Terms}$,
- $\{t\}_k$ where $t \in \mathit{Terms}$ and $k \in \mathit{ATerms}$,
- $\{\!|t|\!\}_{ag}$ where $t \in \mathit{Terms}$ and $ag \in \mathit{Terms}_{\mathit{Agents}}$.

$\mathit{priv}(ag)$ depicts the private key of agent ag, $\langle t_1, t_2 \rangle$ represents the concatenation of terms t_1 and t_2, $\{t\}_k$ models the symmetric encryption of t by atomic key k,

and $\{\!|t|\!\}_{ag}$ stands for the asymmetric encryption of t with the public key of agent ag. As usual we equate the public keys of agents with their names.

Given two terms t_1, t_2, we write $t_1 \sqsubseteq t_2$ if t_1 is a subterm of t_2. We also define a relation $\sqsubseteq_s$ to express when a term t_1 is *source-contained* in a term t_2. This is inductively defined as follows:

- $t \sqsubseteq_s a$ where $a \in \mathit{ATerms}$ iff $t = a$,
- $t \sqsubseteq_s \mathit{priv}(ag)$, iff $t = \mathit{priv}(ag)$ or $t = ag$,
- $t \sqsubseteq_s \langle t_1, t_2 \rangle$ iff $t = \langle t_1, t_2 \rangle$ or $t \sqsubseteq_s t_1$ or $t \sqsubseteq_s t_2$,
- $t \sqsubseteq_s \{t'\}_k$ iff $t = \{t'\}_k$ or $t \sqsubseteq_s t'$,
- $t \sqsubseteq_s \{\!|t'|\!\}_{ag}$ iff $t = \{\!|t'|\!\}_{ag}$ or $t \sqsubseteq_s t'$.

A *message* is a ground, i.e., variable-free term. A *message template* is a term that does not contain any elements of *Atoms*. We denote the set of messages by *Mesg* and the set of message templates by *TMesg*. A *ground substitution* is a function that assigns messages to variables such that the types are preserved.

$$\frac{x \quad y}{\langle x, y \rangle}\ C_{pair} \qquad \frac{\langle x, y \rangle}{x}\ D_{pair_1} \qquad \frac{\langle x, y \rangle}{y}\ D_{pair_2}$$

$$\frac{x \quad y_a}{\{x\}_{y_a}}\ C_{senc} \qquad \frac{\{x\}_{y_a} \quad y_a}{x}\ D_{senc}$$

$$\frac{x \quad y_{ag}}{\{\!|x|\!\}_{y_{ag}}}\ C_{aenc} \qquad \frac{\{\!|x|\!\}_{y_{ag}} \quad \mathit{priv}(y_{ag})}{x}\ D_{aenc}$$

where $x, y \in \mathit{Vars}$, $y_a \in \mathit{AVars}$, and $y_{ag} \in \mathit{Vars}_{\mathit{Agents}}$.

Fig. 1. The Dolev-Yao intruder deduction system $\mathcal{I}_{DY}$

.

Intruder Deduction Capabilities. We assume the deduction capabilities of the standard Dolev-Yao intruder, modelled by the inference system $\mathcal{I}_{\mathcal{DY}}$ depicted in Fig. 1. The rules of inference fall into matching composition and decomposition rules. The composition rules model the intruder's ability to build new messages from messages he has already deduced while the decomposition rules capture when he can decompose deduced messages into their parts.

Let T be a set of terms, and u be a term. We say u is *deducible* from T in $\mathcal{I}_{\mathcal{DY}}$, written $u \in \mathcal{DY}(T)$, iff there is a proof tree of $T \vdash u$ in $\mathcal{I}_{\mathcal{DY}}$. A *proof* of $T \vdash u$ in an inference system $\mathcal{I}$ is a proof tree Π such that:

- Every leaf of Π is labelled with a term v such that $v \in T$.
- For every node labelled with v_0 having n children labelled with $v_1, \ldots, v_n$, there is an instance of an inference rule with conclusion v_0 and premises $v_1, \ldots, v_n$. We say that Π ends with this instance if the node is the root of Π.
- The root is labelled with u.

Let Π be a proof tree of $T \vdash u$. Let n be a node of Π. We write $lab(n)$ for the label of n. We denote the subtree that is rooted in n by $subtree(n)$. Clearly, it is a proof tree of $T \vdash lab(n)$. We denote the root node of Π by $root(\Pi)$, and the set of leaves of Π by $leaves(\Pi)$. We denote by $Concl(\Pi)$ the label of the root of Π, and by $Hyp(\Pi)$ the set of labels of the leaves of Π. If Π only contains instances of decomposition rules we say Π is a *decomposition tree*, and if it only contains instances of composition rules a *composition tree* respectively. The *size* of a proof tree is the number of its nodes. A proof tree of $T \vdash u$ is *minimal* if there is no other proof tree of $T \vdash u$ with size strictly smaller than Π.

Protocol Specifications. A *protocol* is a pair $P = (roles_P, script_P)$, where $roles_P$ is a finite set of *roles*, and $script_P$ is a function that maps every role in $roles_P$ to a *role script*. Given $r \in roles_P$, $script_P(r)$ is a finite sequence

$$p_1 M_1 \; p_2 M_2 \; \ldots \; p_n M_n$$

where $n \geq 1$, and for every $i \in [1, n]$, $p_i \in \{+, -\}$ and M_i is a message template, i.e. a term without any atom. A term prefixed with a '+' is thought to be sent, and a term prefixed with a '−' to be received respectively. For $i \in [1, n]$ define $fresh_r(i)$ to be the set $\{x \in Vars \mid p_i = \text{`}+\text{'} \;\&\; x \sqsubseteq M_i \;\&\; \forall j \in [1, i[, x \not\sqsubseteq M_j\}$. We require the following three axioms:

r1 $p_1 = \text{`}-\text{'}$, and $\exists A \in Vars_{Agents} \exists M_1' \in TMesg.\ M_1 = A \;\vee\; M_1 = \langle A, M_1' \rangle$,
r2 $\forall i \in [1, n]$, $fresh_r(i) \subseteq AVars$, and
r3 $\forall i \in [1, n]$, if $p_i = \text{`}+\text{'}$ then
$$\{M_j \mid j \in [1, i[\;\&\; p_j = \text{`}-\text{'}\} \cup \{priv(A)\} \cup fresh_r(i) \vdash M_i \ ,$$
where A is given as in axiom (r1).

Role scripts will be interpreted as follows. The ownership of the session will be defined by the agent name to be received in the first component of the first message. This is the reason behind Axiom (r1). If a variable appears for the first time in a message pattern to be received then any value that respects the sort of the variable can be matched to it. If a variable appears for the first time in a message pattern to be sent then a fresh value will be assigned to it at this step. Axiom (r2) ensures that only atoms can be freshly generated. The received or freshly generated value is henceforward understood to be bound to the variable. Axiom (r3) ensures that each message to be sent can be assembled from the messages already received, the private key of the owner of the session, and the atoms to be freshly generated at this step. Our interpretation that variables that first appear in a message pattern to be sent are assumed to be freshly generated is no restriction: public session parameters such as agent names can be received in a message to be sent by the intruder. Our use of message templates is no restriction either: constants can be modelled by types with one element or via adapting the definitions so that agents have prior knowledge of constants.

Example 1. The Needham-Schroeder Public Key (NSPK) Protocol [19] is informally described by the message exchange shown on the left below. Formally, it

is specified by $P = (\mathit{roles}_P, \mathit{script}_P)$, where $\mathit{roles}_P = \{A, B\}$, and $\mathit{script}_P(A)$ and $\mathit{script}_P(B)$ are defined as follows.

	A	B
1. $A \longrightarrow B : \{\!\lvert N_A, A \rvert\!\}_B$	$1 \quad -A, B$	$-B, A$
2. $B \longrightarrow A : \{\!\lvert N_A, N_B \rvert\!\}_A$	$2 \quad +\{\!\lvert N_A, A \rvert\!\}_B$	$-\{\!\lvert N_A, A \rvert\!\}_B$
3. $A \longrightarrow B : \{\!\lvert N_B \rvert\!\}_B$	$3 \quad -\{\!\lvert N_A, N_B \rvert\!\}_A$	$+\{\!\lvert N_A, N_B \rvert\!\}_A$
	$4 \quad +\{\!\lvert N_B \rvert\!\}_B$	$-\{\!\lvert N_B \rvert\!\}_B$

Strands and Bundles. We now define protocol executions in terms of the strand space model (e.g. [16]). More precisely, we work with a variation of the strand space model: first, our definition of bundle will not make use of the usual intruder strands but uses deduction trees whose leaves are mapped to output events. Second, we add a total order to the concept of bundle, which means we have available the execution order of events as well as the causal relationship between them. In the following, assume a fixed protocol $P = (\mathit{roles}_P, \mathit{script}_P)$.

A *strand* represents an instantiation of a role script of the protocol or a prefix thereof. (We admit prefixes to be able to model incomplete protocol sessions; a situation that naturally arises in a snaphshot of a protocol execution.) Formally, a *strand* of P is a totally ordered labelled graph $s = (E, \Rightarrow, l)$ where

- $E = \{(s, 1), \ldots, (s, n)\}$, $n > 0$,
- $(s, i) \Rightarrow (s, j)$ iff $j = i + 1$,
- $l : E \to \{+, -\} \times \mathit{Mesg}$, and

there are $r \in \mathit{roles}_P$, a prefix of $\mathit{script}_P(r)$ of the form $p_1 M_1\ p_2 M_2\ \ldots\ p_n M_n$, and a ground substitution σ such that for all $i \in [1, n]$ we have:

S1 $l(s, i) = p_i M_i \sigma$,
S2 if $x \in \mathit{fresh}_r(i)$ then
  (a) for all $j \in [1, i[$, $x\sigma \not\sqsubseteq M_j\sigma$,
  (b) for all $x' \in \mathit{fresh}_r(i)$, $x\sigma = x'\sigma$ implies $x = x'$.

Observe how the two axioms ensure that s can indeed be understood as an instantiation of the partial role script via σ. Axiom (S1) ensures that the signed message is an instance of the respective signed message template. Axiom (S2) guarantees that if an atom is to be freshly generated for message M_i then (a) it does not occur earlier on the strand, and (b) it is distinct from all other atoms freshly generated for message M_i.

We call E the set of *events of* s. Given an event e of s, we call the first component of $l(e)$ the *sign of* e, written $\mathit{sign}(e)$, and the second component the *message of* e, written $\mathit{msg}(e)$. If $\mathit{sign}(e) =$ '+', we call e an *output event*, and if $\mathit{sign}(e) =$ '$-$' an *input event* respectively. We say atom a *originates on event* (s, i) if (s, i) is an output event, $a \sqsubseteq \mathit{msg}(s, i)$, and for all $j \in [1, i[$, $a \not\sqsubseteq \mathit{msg}(s, j)$. In our technical framework we have the following convenient fact:

Proposition 2. *Let s be a strand such that s is a (partial) instance of $r \in \mathit{roles}_P$ via substitution σ. For all events (s, i) of s we have: $x \in \mathit{fresh}_r(i)$ iff $x\sigma$ originates on (s, i).*

We assume a special strand *init*, which consists of a finite set of output events that models the intruder's initial knowledge he has prepared for the protocol run. We assume that he has available at least his own private key, denoted by $priv(i)$, and an atom of each atomic type. By abuse of notation we usually consider this strand as one event *init*.

An *ordered strand space of* P is a pair $\mathcal{S} = (S, <)$ where S is a set of strands of P, and $<$ is a total order on the events of S such that

S1 $init \in S$,
S2 $\Rightarrow \subseteq <$, and
S3 $init < e$ for every event $e \neq init$.

Axiom (S1) models that the intruder is always expected to prepare some initial knowledge. Axiom (S2) expresses that if e' precedes e in a session then e' must have happened before e. Axiom (S3) expresses that the intruder generates all atoms he will use in the attack in advance.

In the context of an ordered strand space $\mathcal{S} = (S, <)$, we denote the set of events of S by E, the set of input events by E_{in}, and the set of output events by E_{out} respectively. Given $e \in E$, the *downwards closure of* e *in* $\mathcal{S}$ is defined by $e\Downarrow = \{e' \mid e' \leq e\}$, and the *strict downwards closure of* e *in* $\mathcal{S}$ by $e\downarrow = \{e' \mid e' < e\}$. When $\mathcal{S}$ is not uniquely determined by the context we also use $e\Downarrow_{\mathcal{S}}$, and $e\downarrow_{\mathcal{S}}$ respectively.

Let $\mathcal{S}$ be an ordered strand space. A *source (proof) tree* wrt $\mathcal{S}$ is a pair (Π, src) where Π is a proof tree, and $src : leaves(\Pi) \to E_{out}$ is a map from the leaf nodes of Π to the output events of $\mathcal{S}$ such that $src(n_l) = e$ implies $lab(n_l) = msg(e)$. Given a message m, we say (Π, src) is a source tree for m wrt $\mathcal{S}$ if $Concl(\Pi) = m$.

A bundle represents a snapshot of a protocol execution. Formally, a *bundle* of P is a tuple $B = (\mathcal{S}, \{(\Pi_e, src_e)\}_{e \in E_{in}})$ where

- $\mathcal{S} = (S, <)$ is an ordered strand space of P, and
- $\forall e \in E_{in}$, (Π_e, src_e) is a source tree for $msg(e)$ wrt $e\downarrow$

such that

B1 every atom a occurring in S has a unique origin in S: there is exactly one event $e \in E$ such that a originates on e.

The required family of source trees ensures that the message of every input event can be deduced by the intruder from the messages of the previous output events (including the intruder's own event *init*.) Axiom (B1) together with Prop. 2 ensures that if an atom is thought to be freshly generated on some strand then on any other strand it has to be received before it can be sent. Given an atom a that occurs in $\mathcal{S}$, define the *origin* of a, denoted by $origin(a)$, to be the unique event on which a originates as guaranteed by Axiom (B1).

Define $\to \subseteq E_{out} \times E_{in}$ by: $e \to e'$ iff there is a leaf $n_l \in \Pi_{e'}$ such that $src_{e'}(n_l) = e$. It is easy to check that $\to \subseteq <$. We denote the relation $\to \cup \Rightarrow$ by $\prec_1$. $\prec_1$ expresses *immediate causality*: if $e \to e'$ then e is an immediate cause of e' due to the intruder deduction causality between messages captured by the

intruder to deduce a message that is then injected by him into a protocol session. If $e \Rightarrow e'$ then e is an immediate cause of e' due to the execution order causality within a protocol session. The transitive closure of $\prec_1$, denoted by $\prec$, is a strict order, which captures *causality*. It is compatible with the execution order $<$: we have $\prec \subseteq <$, and hence, $\prec$ is a strict order.

A *bundle skeleton* of P is an ordered strand space $\mathcal{S}$ of P that can be extended to form a bundle, i.e., there is a family of source trees $\{(\Pi_e, src_e)\}_{e \in E_{in}}$ such that $(\mathcal{S}, \{(\Pi_e, src_e)\}_{e \in E_{in}})$ is a bundle of P.

We carry over our notations E, E_{in}, E_{out}, $e\Downarrow$, and $e\downarrow$ from ordered strand spaces to bundles and bundle skeletons in the obvious way. Moreover, we write $\mathcal{DY}(\mathcal{S})$ and $\mathcal{DY}(B)$ short for $\mathcal{DY}(M)$, where M is the set of messages of output events of $\mathcal{S}$, and B respectively.

The Leakiness Problem. We now define the secrecy problem as formulated by Ramanujam and Suresh (e.g. [22]). We call the problem *leakiness* to avoid confusion with other notions of secrecy. We also slightly generalize it to include leaks of private keys. Informally, a protocol run is considered to be *leaky* if (1) some atom is secret at some intermediate state of the run but known to the intruder at the end of the run, or (2) the private key of some agent other than the intruder is known to the intruder at the end of the run. The *leakiness problem* is then to check whether a protocol has a leaky run.

Let P be a protocol, and $\mathcal{S}$ be a bundle skeleton of P. An atom a *originates secretly* in $\mathcal{S}$ if $a \notin \mathcal{DY}(origin(a)\Downarrow)$. We say an atom a is a *leak* in $\mathcal{S}$ if a originates secretly in $\mathcal{S}$ but $a \in \mathcal{DY}(\mathcal{S})$. We say a private key $priv(ag)$ is a *leak* in $\mathcal{S}$ if $ag \neq i$ but $priv(ag) \in \mathcal{DY}(\mathcal{S})$. $\mathcal{S}$ is *leaky* if there is a leak in $\mathcal{S}$ and *non-leaky* otherwise. These notions carry over to bundles in the usual way.

LEAKINESS:
Given: A protocol P.
Decide: Is there a bundle B of P such that B is leaky?

2 Honest Causality and Well-Founded Protocols

Well-founded protocols are defined to syntactically exclude what is common to all protocols used in undecidability results: that honest information can be propagated unboundedly without the intruder manipulating it. First, we define two core concepts, *honest encryptions* and *source paths*. Based on these concepts we formalize the idea of unmanipulated information propagation in terms of a relation $\prec_h$, called *honest causality*. This will lead us to our definition of *well-founded protocols* We then confirm that for well-founded protocols the *honest causal depth* of each bundle is indeed bounded by the depth of the protocol. Finally, we define the concept of *honest cause components*.

Honest Encryptions and Source Paths. Let M be a set of messages the intruder has available at some stage of a protocol run. We wish to single out

those encryptions that he can deduce but that he cannot analyse nor synthesize. We call them the *honest encryptions wrt* M since they must have come from an honest agent.

Given a set of messages M, an *honest encryption wrt* M is a message $h \in \mathcal{DY}(M)$ such that

1. h is a symmetric encryption $\{m\}_k$ such that $k \notin \mathcal{DY}(M)$, or
2. h is a public key encryption $\{\!|m|\!\}_{ag}$ such that $m \notin \mathcal{DY}(M)$.

We denote the set of honest encryptions wrt M by $HEnc(M)$. Given a strand space S, we write $HEnc(S)$ short for $HEnc(M)$, where $M = \{msg(e) \mid e \in E_{out}\}$.

Let m be a message, and Π be a proof tree. Say m is source-contained in the conclusion of Π. We are interested in tracing back where m originates from in the deduction. We translate this into the concept of *source path*. Given a message m, and a proof tree Π, we define a path π of Π to be an *intermediate source path of* m *in* Π iff

> *Base case:* $\pi = n_r$ such that n_r is the root of Π, and $m \sqsubseteq_s lab(n_r)$, or
> *Inductive case:* $\pi = \pi' n$ such that π' is an intermediate source path of m in Π, and one of the following holds with respect to the last node n' of π' and the rule instance it is conclusion of:

$$\frac{c_1 : m_1 \qquad [c_2 : m_2]}{n' : m'}\ R$$

1. $R = C_{pair}$, n is the left child c_1, and $m \sqsubseteq_s m_1$,
2. $R = C_{pair}$, n is the right child c_2, and $m \sqsubseteq_s m_2$,
3. $R = C_{senc}$ or C_{aenc}, n is the left child c_1, and $m \sqsubseteq_s m_1$,
4. $R = D_{pair_i}$ for $i = 1$, or 2, or
5. $R = D_{senc}$ or D_{aenc}, and n is the left child c_1.

We say π is a *soure path of* m *in* Π if it is a maximal intermediate source path of m in Π.

Proposition 3. *Let m be a message, Π be a proof tree, and π be a source path of m in Π. For all nodes n of π, we have $m \sqsubseteq_s lab(n)$.*

Note that if there are several occurrences of m in $Concl(\Pi)$ then there can be several source paths of m in Π. A source path will trace back m either to a leaf, or to the conclusion of a composition node. Since atoms and private keys cannot be composed they can always be traced back to leaves. Moreover, if m is an honest encryption wrt $Hyp(\Pi)$ then it can also be traced back to leaves since it can neither be composed nor decomposed.

Proposition 4. *Let M be a set of messages, m a message, and Π a proof tree of $M \vdash m$. If $h \in HEnc(M)$ such that $h \sqsubseteq_s m$ then there is a source path π of h in Π such that π ends in a leaf.*

Honest Causality. Let P be a protocol, and let $B = (S, <, \{(\Pi_e, src_e)\}_{e \in E_{in}})$ be a bundle of P. We define a relation $\rightarrow_h \subseteq E_{out} \times E_{in}$ as follows: $e' \rightarrow_h e$ iffthere is $h \in HEnc(e\downarrow)$ such that there is a source path of h in Π_e ending in a leaf node n_l with $src_e(n_l) = e'$.

We denote the relation $\rightarrow_h \cup \Rightarrow$ by $\prec_h^1$. $\prec_h^1$ expresses that information is propagated unmanipulated directly from one protocol event to another. If $e' \Rightarrow e$ this is so because e' precedes e in a protocol session. If $e' \rightarrow_h e$ this is so because an encryption is passed from e' to e in unmanipulated form: because it can neither be analysed nor synthesized by the intruder at that point. The transitive closure of $\prec_h^1$, denoted by $\prec_h$, is a strict order, which captures unmanipulated information propagation. We call it *honest causality*. It is straightforward to check that $\prec_h$ is a strict order.

Given an event $e \in E$, we define the *honest causal depth of* e, written $depth_{\prec_h}(e)$, inductively as follows: if e is minimal wrt $\prec_h$ then $depth_{\prec_h}(e) = 0$; otherwise $depth_{\prec_h}(e) = 1 + \max\{depth_{\prec_h}(e') \mid e' \prec_h^1 e\}$. The *honest causal depth* of B, denoted by $depth_{\prec_h}(B)$, is given by $\max\{depth_{\prec_h}(e) \mid e \in E\}$.

Well-Founded Protocols. We now design a protocol class, which satisfies: given a protocol P of the class, there is $n \in \mathsf{Nat}$ such that for every bundle B of P, $depth_h(B)$ is bounded by n. We base the definition on a preorder on protocol positions that captures *potential* unmanipulated information propagation.

Let P be a protocol. We define the set of *protocol positions* of P by $Pos = \{(r, i) \mid r \in roles_P \ \&\ 1 \leq i \leq |script_P(r)|\}$. Define the *output positions* of P by $Pos_{out} = \{(r, i) \in Pos \mid p_i = \text{`+'} \text{ in } script_P(r)\}$, and the *input positions* by $Pos_{in} = \{(r, i) \in Pos \mid p_i = \text{`}-\text{'} \text{ in } script_P(r)\}$ respectively.

First, define a relation $\Rightarrow_P \subseteq Pos \times Pos$ by $(r, i) \Rightarrow_P (r', j)$ iff $r = r'$ and $i < j$. Second, define a relation $\rightarrow_P \subseteq Pos_{out} \times Pos_{in}$, which describes when an enryption sent can possibly match an encryption received during a protocol run: $p \rightarrow_P p'$ iff there are substitutions σ and σ', and an encryption m_e such that $m_e \sqsubseteq msg(p)\sigma$, and $m_e \sqsubseteq msg(p')\sigma'$.

We denote the relation $\rightarrow_p \cup \Rightarrow_p$ by $\prec_p^1$. $p \prec_p^1 p'$ expresses that information might be passed in a protected manner directly from an instance of p to an instance of p'. If $p \Rightarrow_p p'$ then this is so because p precedes p' in a role script. If $p \rightarrow_p p'$ this is so because an encryption, which is possibly neither analysable nor composable by the intruder, might be sent from an instance of p to an instance of p'. The reflexive and transitive closure of $\prec_p^1$, denoted by $\preceq_p$, is a preorder that captures *potential* unmanipulated information propagation.

Definition 5. *We say a protocol P is* well-founded *iff $\prec_p^1$ is acyclic, or $\preceq_p$ is a partial order equivalently.*

Let P be a well-founded protocol. Given a position $p \in Pos$, we define the *depth of* p, written $depth_{\prec_p}(p)$, inductively as follows: if p is minimal wrt to $\prec_p$ then $depth_{\prec_p}(p) = 0$; otherwise $depth_{\prec_p}(p) = 1 + \max\{depth_{\prec_p}(p') \mid p' \prec_p^1 p\}$. The *depth of protocol* P, denoted by $depth(P)$, is given by $\max\{depth_{\prec_p}(p) \mid p \in Pos\}$.

Example 6. Recall the NSPK protocol from Example 1. It is easy to check that $\prec_p^1$ is acyclic, and hence, the NSPK protocol is well-founded. Moreover, observe that it has depth 6.

Example 7. The Woo and Lam protocol (c.f. [6]) is a flawed authentication protocol where B wants to verify that A is present with the help of a server S.

$$
\begin{array}{l}
1.\ A \longrightarrow B : A \\
2.\ B \longrightarrow A : N_B \\
3.\ A \longrightarrow B : \{N_B\}_{sh(A,S)} \\
4.\ B \longrightarrow S : \{A, \{N_B\}_{sh(A,S)}\}_{sh(B,S)} \\
5.\ S \longrightarrow B : \{N_B\}_{sh(B,S)}
\end{array}
$$

Formally, the protocol is specified by the following role scripts:[1]

	A	B	S
1	$-A, B, S$	$-B, A, S$	$-S, A, B$
2	$+A$	$-A$	$-\{A, \{N_B\}_{sh(A,S)}\}_{sh(B,S)}$
3	$-N_B$	$+N_B$	$+\{N_B\}_{sh(B,S)}$
4	$+\{N_B\}_{sh(A,S)}$	$-X$	
5		$+\{A, X\}_{sh(B,S)}$	
6		$-\{N_B\}_{sh(B,S)}$	

The Woo and Lam protocol is not well-founded: e.g., we have $(B, 4) \Rightarrow_p (B, 5) \rightarrow_p (B, 4)$, and $(S, 2) \Rightarrow_p (S, 3) \rightarrow_p (S, 2)$. The first situation is an example of how blind copies always cause $\prec_p^1$ to be cyclic.

Example 8. The Otway-Rees protocol (c.f. [6]) establishes a shared secret between two agents with the help of a trusted server.

$$
\begin{array}{l}
1.\ A \longrightarrow B : A, B, \{N_A, A, B\}_{sh(A,S)} \\
2.\ B \longrightarrow S : A, B, \{N_A, A, B\}_{sh(A,S)}, \{N_B, A, B\}_{sh(B,S)} \\
3.\ S \longrightarrow B : \{N_A, K_{AB}\}_{sh(A,S)}, \{N_B, K_{AB}\}_{sh(B,S)} \\
4.\ B \longrightarrow A : \{N_A, K_{AB}\}_{sh(A,S)}
\end{array}
$$

Formally, it has the following role scripts:

	A	B	S
1	$-A, B, S$	$-B, A, S$	$-S, A, B$
2	$+A, B, \{N_A, A, B\}_{sh(A,S)}$	$-A, B, X$	$-A, B, \{N_A, A, B\}_{sh(A,S)}, \{N_B, A, B\}_{sh(B,S)}$
3	$-\{N_A, K_{AB}\}_{sh(A,S)}$	$+A, B, X, \{N_B, A, B\}_{sh(B,S)}$	$+\{N_A, K_{AB}\}_{sh(A,S)}, \{N_B, K_{AB}\}_{sh(B,S)}$
4		$-Y, \{N_B, K_{AB}\}_{sh(B,S)}$	
5		$+Y$	

It is easy to see that $\prec_p^1$ is cyclic due to blind copies: e.g., we have $(B, 2) \Rightarrow_p (B, 3) \rightarrow_p (B, 2)$ by substituting any encryption for X. Note that in this example blind copies are merely relayed from an input concatenation to an output concatenation, and the Dolev-Yao intruder could have provided the relaying himself. Hence, the following simplified formalization is adequate with respect to most analysis problems (including leakiness).

[1] Long-term shared keys $sh(A, B)$ can be added to our framework analogously to public keys.

	A	B	S
1	$-A, B, S$	$-B, A, S$	$-S, A, B$
2	$+A, B, \{N_A, A, B\}_{sh(A,S)}$	$-A, B$	$-A, B, \{N_A, A, B\}_{sh(A,S)}, \{N_B, A, B\}_{sh(B,S)}$
3	$-\{N_A, K_{AB}\}_{sh(A,S)}$	$+A, B, \{N_B, A, B\}_{sh(B,S)}$	$+\{N_A, K_{AB}\}_{sh(A,S)}, \{N_B, K_{AB}\}_{sh(B,S)}$
4		$-\{N_B, K_{AB}\}_{sh(B,S)}$	

It is easy to check that this transformed version is well-founded.

The examples are representative for other well-known authentication and key establishment protocols of the Clark/Jacob library [6] and the collection of Boyd and Mathuria [4]. Protocols for authentication and key establishment without a trusted third party are typically well-founded. For example, this also includes the corrected NSPK by Lowe, the ISO/IEC 9798-2 three pass mutual authentication protocol, and the revised Andrew protocol of Burrows et al. Protocols with a trusted third party typically involve that the server sends a ticket to one of the agents, who then passes it on to another agent. Since tickets formally lead to blind copies such protocols are not well-founded. However, many can be transformed into equivalent well-founded protocols similarly to the Otway-Rees protocol in Example 8. This includes Kerberos Version 5, Yahalom, and Bauer-Berson-Feiertag. Other protocols send the ticket in an encryption and will not allow this transformation. Examples are Needham-Schroeder Shared Key, and the Denning-Sacco protocol. Interestingly, it seems difficult to find real protocols without flaws that fail to be well-founded without this being due to blind copies.

The Characteristic. We are now ready to establish the characteristic of well-founded protocols P: *for every bundle B of P, the honest causal depth of B is bounded by the depth of P.* This follows from: More precisely, for every event e of B, the honest causal depth of e is bounded by the depth of the protocol position that is instantiated by e.

Lemma 9. *Let P be a well-founded protocol, and let B be a bundle of P. For all $e \in E$ we have: $depth_{\prec_h}(e) \leq depth_{\prec_p}(pos(e))$.*

Theorem 10. *Let P be a well-founded protocol. For all bundles B of P we have: $depth_{\prec_h}(B) \leq depth(P)$.*

Honest Cause Components. Let P be a protocol in general, and let $B = (S, <, \{\Pi_e, src_e\}_{e \in E_{in}})$ be a bundle of P. Given $e' \in E$, define the *honest cause set* of e' by $e' \Downarrow_h = \{e'' \mid e'' \preceq_h e'\}$, and the *strict honest cause set* of e' by $e' \downarrow_h = \{e'' \mid e'' \prec_h e'\}$ respectively. When B is not uniquely determined by the context we also use $e' \Downarrow_h^B$ and $e' \downarrow_h^B$. The *honest cause component induced by e'* is defined by $(S \restriction E_C, <\restriction E_C, \{\Pi_e, src_e\}_{e \in E_{in} \restriction E_C})$ where $E_C = e' \Downarrow_h \cup \{init\}$.

3 Well-Structured Source Trees and Bundles

This section is about *well-structured* source trees and bundles. First we introduce two basic concepts, *bricks* and the *event of first deducibility* of a message. Then we provide the definition based on these concepts. We show that every bundle can

be transformed into a well-structured bundle with the same underlying skeleton. Finally, we prove two structural lemmas: the BWS Lemma, and the WS Lemma respectively. The latter will allow us to transform honest cause components into bundles in Section 4. In the following, fix a protocol P.

Basic Concepts. Given a set of messages M, we wish to capture those messages that form the smallest units the intruder has available to build new messages. Such *bricks* are deducible atoms, private keys, and encryptions that can neither be analysed nor synthesized, i.e. honest encryptions. Formally, a *brick wrt* M is a message $b \in \mathcal{DY}(M)$ such that (1) b is an atom, (2) b is a private key, or (3) b is an honest encryption wrt M. We denote the *set of bricks wrt* M by $Bricks(M)$. Given an ordered strand space $\mathcal{S}$, we write $Bricks(\mathcal{S})$ short for $Bricks(M)$ where $M = \{msg(e) \mid e \in E_{out}\}$. We use $Bricks(e\downarrow)$, $Bricks(e\Downarrow)$, and $Bricks(B)$ with the analogous interpretation. Note that $Bricks(M)$ is not the same as the fringe of $analz(M)$: a public key encryption might be synthesizable by the intruder but not analyzable.

Let $\mathcal{S} = (S, <)$ be an ordered strand space. Given a message $m \in \mathcal{DY}(\mathcal{S})$, we single out the event e at which m becomes deducible for the first time wrt $<$. Formally, the *event of first deducibility of* m *wrt* $\mathcal{S}$, denoted by $efd_{\mathcal{S}}(m)$, is defined to be the event $e \in E$ such that $m \in \mathcal{DY}(e\Downarrow)$, and e is minimal wrt $<$, i.e. for all other events $e' \in E$ such that $m \in \mathcal{DY}(e'\Downarrow)$, $e < e'$.

Definition and Existence. The general idea behind well-structured bundles is this: if the intruder needs to deduce a message m at the current stage of the protocol run then he will compose m from units that are bricks at this stage; and he will deduce each thus employed brick b by a decomposition using the message of the event, say e, at which b has first become deducible in the protocol run, and units that have been available as bricks at the stage before e. And this is continued in an inductive fashion.

Assume an ordered strand space $\mathcal{S} = (S, <)$. We first define when a source tree is *brick-well-structured* wrt $\mathcal{S}$. This is the case when its conclusion, say b, is a brick wrt $\mathcal{S}$, and b is the result of a minimal decomposition of the message of $efd_{\mathcal{S}}(b)$, say e, and elements of $Bricks(e\downarrow)$. The latter are deduced by source trees that are brick-well-structured wrt $e\downarrow$ in an inductive fashion.

Let (Π, src) be a source tree wrt $\mathcal{S}$. Formally, we define when (Π, src) is *brick-well-structured wrt* $\mathcal{S}$ by induction on the size of $\mathcal{S}$ (i.e. the number of events of $\mathcal{S}$).
Base case $|\mathcal{S}| = 0$: There are no source trees wrt such $\mathcal{S}$.
Inductive case $|\mathcal{S}| > 0$: (Π, src) is *brick-well-structured wrt* $\mathcal{S}$ iff, setting $b = Concl(\Pi)$, $e = efd_{\mathcal{S}}(b)$, and $m = msg(e)$,

- there is a minimal proof tree Π_d of $Bricks(e\downarrow) \cup \{m\} \vdash b$ with leaves $l_1 : m, \ldots, l_j : m, l'_1 : b_1, \ldots, l'_k : b_k$ for some $j > 0$, $k \geq 0$, and
- for all $i \in [1, k]$, there is a brick-well-structured source tree (Π_i, src_i) with $Concl(\Pi_i) = b_i$ wrt $e\downarrow$,

such that

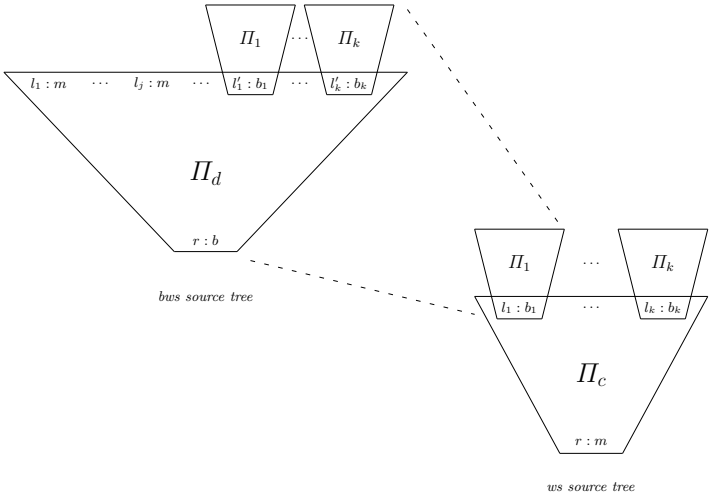

Fig. 2. Well-structured source trees

1. $b \in \mathit{Bricks}(\mathcal{S})$,
2. Π is the composition of Π_d, $\Pi_1, \ldots, \Pi_k$ by replacing for each $i \in [1, k]$ the leaf l'_i by the proof tree Π_i, and
3. $\mathit{src} = \{(l_i, e) \mid i \in [1, j]\} \cup \bigcup\{\mathit{src}_i \mid i \in [1, k]\}$.

We say a source tree (Π, src) is brick-well-structured for b wrt $\mathcal{S}$ if it is brick-well-structured wrt $\mathcal{S}$ and $\mathit{Concl}(\Pi) = b$.

A source tree with conclusion m is *well-structured wrt* $\mathcal{S}$ if it deduces m in two stages: first, it composes m from bricks wrt $\mathcal{S}$ in a minimal way; second, it deduces each employed brick b by a source tree that is *brick-well-structured wrt* $\mathcal{S}$. Let (Π, src) be a source tree wrt $\mathcal{S}$. Formally, we say (Π, src) is *well-structured wrt* $\mathcal{S}$ iff, setting $m = \mathit{Concl}(\Pi)$,

- there is a minimal proof tree Π_c of $\mathit{Bricks}(\mathcal{S}) \vdash m$ with leaves $l_1 : b_1, \ldots, l_k : b_k$ for some $k > 0$, and
- for all $i \in [1, k]$, there is a brick-well-structured source tree (Π_i, src_i) for b_i wrt $\mathcal{S}$

such that

1. Π is composed of Π_c, Π_1, …, Π_k by replacing for each $i \in [1, k]$ the leaf l_i by the proof tree Π_i, and
2. $\mathit{src} = \bigcup\{\mathit{src}_i \mid i \in [1, k]\}$.

We say a source tree (Π, src) is well-structured for m wrt $\mathcal{S}$ if it is well-structured wrt $\mathcal{S}$ and $\mathit{Concl}(\Pi) = m$.

Proposition 11. *Let $\mathcal{S}$ be an ordered strand space, and $m \in \mathcal{DY}(\mathcal{S})$. Then there is a well-structured source tree for m wrt $\mathcal{S}$.*

Let $B = (\mathcal{S}, \{(\Pi_e, src_e)\}_{e \in E_{in}})$ be a bundle. B is *well-structured* if for every $e \in E_{in}$, (Π_e, src_e) is a well-structured source tree wrt $e{\downarrow}$.

Theorem 12. *For every bundle B, there is a well-structured bundle B' such that B' has the same underlying skeleton as B.*

In the following, we abbreviate brick-well-structured by *bws*, and well-structured by *ws* respectively. Moreover, we make use of the fact that it can be proved that bws trees are decomposition trees, and that Π_c in ws trees is a composition tree.

The BWS Lemma. We now prove a characteristic lemma about bws source trees, called the *BWS Lemma*. Assume an ordered strand space $\mathcal{S}$, and a bws source tree (Π, src) wrt $\mathcal{S}$. Given a node n of Π such that n is labelled by a brick $b \in Bricks(\mathcal{S})$, it is easy to trace back from which leaf b stems from. This is so because Π is a decomposition tree.

Proposition 13. *Let n be a node of Π such that n is labelled by a brick $b \in Bricks(\mathcal{S})$. There is exactly one source path of b in subtree(n). Moreover, it ends in a leaf. We call it srcpath$\uparrow_{\Pi}(n)$.*

The BWS Lemma says: either we can trace back the source of b to the event of first deducibility of b wrt $\mathcal{S}$, or we can exhibit a leak in $\mathcal{S}$. The latter is the case when b exists wrapped by an encryption at some stage of the run, but b becomes first deducible only at a later stage when the encryption can be decrypted by the intruder. We translate this into the tool of *leak witness situations*.

Definition 14. *Let $\mathcal{S}$ be an ordered strand space, and $e \in E$. We say a proof tree Π contains a* leak witness situation wrt $\mathcal{S}$ and e *iff Π contains*

1. *an instance of rule D_{senc}*

$$\frac{n_l : \{m\}_k \quad n_r : k}{n_p : m}\ D_{senc}$$

   such that $\{m\}_k \in HEnc(e{\downarrow})$ and $k \in \mathcal{DY}(e{\Downarrow})$, or
2. *an instance of rule D_{aenc}*

$$\frac{n_l : \{\!|m|\!\}_{ag} \quad n_r : priv(ag)}{n_p : m}\ D_{aenc}$$

   such that $\{\!|m|\!\}_{ag} \in HEnc(e{\downarrow})$ and $priv(ag) \in \mathcal{DY}(e{\Downarrow})$ respectively.

We call n_l the cipher node *and n_r the* d-key node *of the leak witness situation.*

Proposition 15. *Let $\mathcal{S}$ be a bundle skeleton, (Π, src) be a source tree wrt $\mathcal{S}$, and $e \in E$. If Π contains a leak witness situation wrt $\mathcal{S}$ and e then $\mathcal{S}$ is leaky. More precisely, the label of the d-key node is a leak at e.*

Lemma 16 (BWS Lemma). *Let $\mathcal{S}$ be a bundle skeleton, and let (Π, src) be a bws source tree wrt $\mathcal{S}$. For each node n of Π such that n is labelled by a brick $b \in Bricks(\mathcal{S})$ one of the following situations holds:*

1. *srcpath*$\uparrow_{\Pi}(n)$ *ends in a leaf* n_l *such that* $src(n_l) = efd_{\mathcal{S}}(b)$, *or*
2. *srcpath*$\uparrow_{\Pi}(n)$ *passes through a node* n' *such that* n' *is the cipher node of a leak witness situation wrt* $\mathcal{S}$ *and* $efd_{\mathcal{S}}(b)$.

The proof of the lemma proceeds by case analysis. Assume Π is of the format of Fig. 2, and set $\pi = srcpath_{\Pi}(n)$. There are three cases to consider: (1) π is entirely contained in Π_d and does not intersect with the roots of the Π_i; then the first situation can be shown to apply. (2) π starts in Π_d but passes through a Π_i. Then we can exhibit a leak witness situation at the transition from Π_d into Π_i, and the second situation applies. (3) π is entirely contained in one of the Π_i. Then we can argue by induction hypothesis.

By Prop. 15 a leak witness situation indeed implies a leak. Then we further obtain the corollary below, where the following fact yields a special case.

Proposition 17. *Let* $\mathcal{S}$ *be a non-leaky bundle skeleton. For all atoms* $a \in \mathcal{DY}(\mathcal{S})$ *we have:* $efd_{\mathcal{S}}(a) = origin_{\mathcal{S}}(a)$.

Corollary 18. *Let* $\mathcal{S}$ *be a* non-leaky *bundle skeleton, and let* (Π, src) *be a bws source tree wrt* $\mathcal{S}$. *For each node* n *of* Π *such that* n *is labelled by a brick* $b \in Bricks(\mathcal{S})$, *we have:* *srcpath*$\uparrow_{\Pi}(n)$ *ends in a leaf* n_l *such that* $src(n_l) = efd_{\mathcal{S}}(b)$. *Moreover, if* b *is an atom then* $src(n_l) = origin_{\mathcal{S}}(a)$.

The WS Lemma. Let B be a ws bundle, and let (Π, src) be the source tree of some input event e of B. Assume Π is of the format of Fig. 2. The WS Lemma gives us insights about the source events of the leaves of Π.

Let n_l be a leaf of Π. Consider the largest path π that traces all ancestor nodes of n_l such that the first and only the first node of π is labelled by a brick wrt $e\downarrow$. Since n_l belongs to some bws source tree, say Π_i, the path π clearly exists. Call its first node n, and let the label of n be b. Since Π_i is a decomposition tree, π is a source path of b in $subtree(n)$. Hence, it coincides with *srcpath*$\uparrow_{\Pi_i}(n)$.

Proposition 19. *Let* n_l *be a leaf of* Π, *and let* Π_i *be the bws source tree within* Π *that* n_l *belongs to.* n_l *has an ancestor node* n *such that* n *is labelled by a brick* $b \in Bricks(e\downarrow)$ *and* *srcpath*$\uparrow_{\Pi_i}(n)$ *ends in* n_l. *We call* n *the special ancestor of* n_l, *denoted by* $anc(n_l)$.

Using the existence of $anc(n_l)$ we obtain the following statement about the source event of n_l:

Lemma 20 (WS Lemma). *Let* B *be a ws bundle, and let* (Π, src) *be the source tree of some input event* e *of* B. *Let* n_l *be a leaf of* Π, *and let* b *be the label of* $anc(n_l)$. *We have:*

1. *If* b *is an atom and* B *is non-leaky then* $src(n_l) = origin_B(a)$.
2. *If* b *is a private key and* B *is non-leaky then* $b = priv(i)$, *and* $src(n_l) = init$.
3. *If* $b \in HEnc(e\downarrow)$ *then* $src(n_l) \prec_h e$.

The first case is a consequence of Corollary 18. The second case is immediate. To see the third case assume $b \in HEnc(e\downarrow)$ and observe that n must be the root of Π_i (where Π_i is given as in Prop. 19). This is so since Π_i is a decomposition tree (hence b cannot be the left child of a decryption node), and we do not consider compositional keys (hence b cannot be the right child of a decryption node). Consider the path that traces all ancestor nodes of n in Π_c from the root. Since Π_c is a composition tree this path must be a source path of b. If we combine it with $srcpath_{\Pi_i}(n)$ we obtain a source path of b in Π that ends in n_l, and hence by definition $src(n_l) \rightarrow_h e$.

4 Transforming Honest Cause Components into Bundles

The main result of this section is this: given a ws bundle B, if B is minimal leaky then B coincides with the honest cause component induced by the last event of B. We achieve this as follows. Based on the WS Lemma we show how honest cause components of non-leaky ws bundles can be transformed into bundles. The transformation also works for minimal leaky bundles and preserves leakiness. Then by a minimality argument the main result stated above is immediate. In the following, fix a protocol P.

Let B be a non-leaky ws bundle, and C be an honest cause component of B. The only reason why, in general, C is not a bundle is that leaves in the source trees of C might have their sources outside of E_C, i.e. in $E_B \setminus E_C$. Let's take a closer look at this situation. Consider a source tree (Π_e, src_e) of C, and a leaf n_l of Π_e such that $src_e(n_l) \notin E_C$. By Prop. 19 we know that n_l has a special ancestor node n that is labelled by a brick b of $e\downarrow_B$. By Lemma 20 we further obtain that b must be an atom that does *not* originate on an event in C: if b is a private key then $src_e(n_l) = init$, which implies $src_e(n_l) \in E_C$, a contradiction; if $b \in HEnc(e\downarrow)$ then $src_e(n_l) \prec_h e$, which also implies $src_e(n_l) \in E_C$, a contradiction; if b is an atom then $src_e(n_l) = origin_B(a)$, and hence $origin_B(a) \notin E_C$ by our assumption $src_e(n_l) \notin E_C$.

Lemma 21. *Let B be a non-leaky ws bundle, and C be an honest cause component of B. Let (Π_e, src_e) be a source tree for some input event e of C. Then for all leaves n_l of Π_e with $src_e(n_l) \notin E_C$ we have: $anc(n_l)$ is labelled by an atom a with $origin_B(a) \notin E_C$.*

Since for such atoms we do not need to respect unique origination constraints wrt E_C we can substitute them by intruder atoms. Thereby we can transform C into a bundle. For the transformation we first define the concept of atom substitution. An *atom substitution* is a function $\alpha : A \rightarrow A'$ such that $A, A' \subseteq Atoms$ and the map preserves atomic types. We generalize atom substitutions α to proof trees and strands in the obvious way. Given a proof tree Π we write $\Pi\alpha$ for the result of applying α to all terms of Π. Given a strand s, we write $s\alpha$ for the result of applying α to all terms of s.

In Fig. 3 we provide the algorithm, $PruneSubst(C)$. We first define an atom substitution that substitutes every atom of C that does not originate in C by an

PruneSubst(Π, src, α)

1. Traverse Π upwards from the root and for each encountered node n do:
   (a) if $lab(n) = a$ for some $a \in domain(\alpha)$ then do:
      i. $\Pi := \Pi \setminus \{subtree(c) \mid c \text{ is a child of } n\}$
      ii. $src := src \setminus \{(n, e') \mid (n, e') \in src \text{ for any } e'\}$
      iii. $src := src \cup \{(n, init)\}$
2. $\Pi := \Pi\alpha$

PruneSubst(C)

1. $\alpha = \{(a, c) \mid a$ is an atom of C such that $origin_B(a) \notin E_C$
   & c is an atom known to the intruder of the same type as $a\}$.
2. For every input event e of C do *PruneSubst*(Π_e, src_e, α)
3. For every strand s of C do $s := s\alpha$

Fig. 3. Algorithm *PruneSubst*

intruder atom. Then for every proof tree of an input event $e \in E_C$ we proceed as follows. We traverse the proof tree from the root upwards until we hit a node that is labelled by an atom to be substituted. We prune the tree so that such nodes become leaves. We assign *init* to be the source of such new leaves. We then apply the atom substitution to the entire tree. Thereby *init* provides indeed an appropriate source for the new leaves. Finally, we apply the substitution to the strands of C. By Lemma 21 we thereby eliminate all leaves that do not have their source in E_C. Moreover, since we only substitute atoms that do not originate in E_C unique origination constraints are not compromised. Together this ensures that the result is a bundle.

Theorem 22. *Let B be a non-leaky ws bundle, and let C be an honest cause component of B. Then PruneSubst(C) is a bundle.*

Since minimal leaky bundles are non-leaky bundles with one additional output event the transformation carries over to minimal leaky bundles. Moreover, the transformation is not too strong in that it preserves leakiness. This is non-trivial to show and requires a variant of Lemma 21.

Theorem 23. *Let B be a minimal leaky ws bundle, and let C be the honest cause component of the last event of B. Then PruneSubst(C) is a leaky bundle. (Indeed, it is also minimal leaky.)*

Now the main result is immediate: given B and C as above, B must coincide with C. Otherwise there is an event in B that is not in C. Then *PruneSubst*(C) yields a leaky bundle that is smaller than B, a contradiction to the assumption that B is minimal leaky.

Corollary 24. *Let B be a minimal leaky ws bundle, and let C be the honest cause component of the last event of B. Then $B = C$.*

5 Main Result and Further Research

Theorem 25. LEAKINESS *is decidable for well-founded protocols.*

To decide LEAKINESS we only need to check whether there exists a minimal leaky bundle. By Theorem 12 it is sufficient to only consider minimal leaky *well-structured* bundles. By Corollary 24 their size is bounded by that of the honest cause components of ws bundles. For well-founded protocols the latter are bounded. This follows from Theorem 10, and the fact that the message size is bounded: we work with typed messages, and the definition of well-foundedness excludes blind copies. Altogether this means we only have to consider bundles of bounded size. As usual we can then work with a fixed alphabet, guess a candidate bundle, and check whether it is indeed a minimal leaky bundle.

For technical ease we have proved the result for a simple message algebra with symmetric and asymmetric encryption. It is straightforward to lift all concepts and proofs to include other cryptographic operations such as digital signatures, MACs, and cryptographic hash functions. Rather than only defining honest encryptions one can work with honest ciphertexts that fall into the corresponding subcases. Moreover, by the approach of Arapinas in [2] the decidability result can be lifted to an untyped message algebra.

Non-trivial extensions include generalizing the result to allow compositional keys and a restricted form of ciphertext forwarding that cannot be dealt with by the simple transformation suggested for Example 8. Now that the proof is in place we can also investigate which standard algebraic theories [8] can be plugged into the framework. One will only have to check that certain properties of honest ciphertexts are satisfied. That this is possible is also indicated by [11], where similar ideas are independently used to obtain decidability in the context of Diffie-Hellmann protocols.

We see our result on LEAKINESS as a major step towards obtaining decidability for a class of standard authentication properties. We hope that some of the insights of Dougherty and Guttman in [11] might help to achieve this. Their security problem is expressed as geometric sequents, and hence, is much more general than LEAKINESS; on the other hand, their proof relies on the fact that there are no temporary secrets. This is naturally given since their class of Diffie-Hellman protocols does not include encryption. Hence, a combination of the techniques might cover a wide class of protocols and security properties.

Acknowledgements. The author is very grateful to Joshua Guttman who has shepherded this paper. His comments and feedback have made this a much improved paper. The author would also like to thank the anonymous referees for their valuable comments.

References

1. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. IEEE Trans. Softw. Eng. 22(1), 6–15 (1996)
2. Arapinis, M., Duflot, M.: Bounding messages for free in security protocols. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 376–387. Springer, Heidelberg (2007)

3. Blanchet, B., Podelski, A.: Verification of Cryptographic Protocols: Tagging Enforces Termination. Theoretical Computer Science 333(1-2), 67–90 (2005), Special issue FoSSaCS 2003
4. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer (2003)
5. Chrétien, R., Cortier, V., Delaune, S.: Typing messages for free in security protocols: The case of equivalence properties. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 372–386. Springer, Heidelberg (2014)
6. Clark, J., Jacob, J.: A survey of authentication protocol literature: Version 1.0 (1997)
7. Comon-Lundh, H., Cortier, V., Zălinescu, E.: Deciding security properties for cryptographic protocols. application to key cycles. ACM Trans. Comput. Logic 11(9), 9:1–9:42 (2010)
8. Cortier, V., Delaune, S., Lafourcade, P.: A survey of algebraic properties used in cryptographic protocols. Journal of Computer Security 14(1), 1–43 (2006)
9. Dolev, D., Yao, A.C.-C.: On the security of public key protocols (extended abstract). In: FOCS, pp. 350–357 (1981)
10. Dolev, S., Even, S., Karp, R.M.: On the security of ping-pong protocols. Inform. and Control 55(1-3), 57–68 (1982)
11. Dougherty, D., Guttman, J.: Decidability for lightweight diffie-hellman protocols. In: CSF 2014, pp. 217–231. IEEE Computer Society (2014)
12. Durgin, N., Lincoln, P., Mitchell, J., Scedrov, A.: Multiset rewriting and the complexity of bounded security protocols. J. of Computer Security 12(2), 247–311 (2004)
13. Even, S., Goldreich, O.: On the security of multi-party ping-pong protocols. In: Symposium on the Foundations of Computer Science, pp. 4–39. IEEE Computer Society (1983)
14. Fröschle, S.: From Security Protocols to Security APIS: Foundations and Verification. To appear in the Information Security and Cryptography series of Springer
15. Fröschle, S.: On well-founded security protocols. In: Joint Workshop on Foundations of Computer Security and Formal and Computational Cryptography (FCS-FCC 2014) (2014)
16. Guttman, J.D., Thayer, F.J.: Authentication tests and the structure of bundles. Theor. Comput. Sci. 283(2), 333–380 (2002)
17. Heintze, N., Tygar, J.D.: A model for secure protocols and their compositions. IEEE Transactions on Software Engineering 22, 2–13 (1996)
18. Lowe, G.: Towards a completeness result for model checking of security protocols. Journal of Computer Security 7(1), 89–146 (1999)
19. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM 21(12), 993–999 (1978)
20. Ramanujam, R., Suresh, S.P.: A decidable subclass of unbounded security protocols. In: WITS 2003, pp. 11–20 (2003)
21. Sarukkai, S., Suresh, S.P.: Tagging makes secrecy decidable with unbounded nonces as well. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 363–374. Springer, Heidelberg (2003)
22. Ramanujam, R., Suresh, S.P.: Decidability of context-explicit security protocols. Journal of Computer Security 13(1), 135–165 (2005)
23. Rusinowitch, M., Turuani, M.: Protocol insecurity with finite number of sessions is NP-complete. In: CSFW 2001, pp. 174–187. IEEE Computer Society (2001)