Sessions and Separability in Security Protocols*

Marco Carbone
IT University of Copenhagen
carbonem@itu.dk

Joshua D. Guttman
Worcester Polytechnic Institute
guttman@wpi.edu

Abstract. Despite much work on sessions and session types in non-adversarial contexts, session-like behavior given an active adversary has not received an adequate definition and proof methods. We provide a syntactic property that guarantees that a protocol has session-respecting executions. Any uncompromised subset of the participants are still guaranteed that their interaction will respect sessions. A protocol transformation turns any protocol into a session-respecting protocol.
We do this via a general theory of separability. Our main theorem applies to different separability requirements, and characterizes when we can separate protocol executions sufficiently to meet a particular requirement. This theorem also gives direct proofs of some old and new protocol composition results. Thus, our theory of separability appears to cover protocol composition and session-like behavior within a uniform framework, and gives a general pattern for reasoning about independence.

Keywords. Sessions, Security Protocols, Strand Spaces

1 Introduction

A transaction or protocol respects sessions if the local runs of the individual participants always match up globally in a compatible way. When one participant receives any message in a session σ, it should have been sent by another participant acting *within the same session* σ. Session-respecting behavior is often studied using *session types* [21,22]. However, most work in this tradition studies sessions within a benign execution environment.

We adapt those ideas to environments containing active adversaries, who may control the medium of communication [28,15,4]. We define session-respecting behavior in an adversarial environment, offering syntactic conditions that ensure a protocol's behavior respects sessions. We exhibit a transformation that, given any protocol, yields one with session-respecting behavior.

Our central idea is *separability.* In an execution, an adversary may receive a message from one session and deliver it, or its fragments, into another session. In this case, we would like to *separate* the sessions by removing the connection that the adversary has created. Separability means we can do this, possibly applying a renaming to one of them so that they involve different parameters. Although

* Carbone thanks the Danish Agency for Science, Technology and Innovation, and Guttman thanks the US National Science Foundation under grant CNS-0952287. Extended version; shorter version appeared in POST 2013.

the adversary can create connections between different session, these connections are inessential. They can be removed modulo renaming. Hence, anything the adversary can achieve in a session, he can also do without relying on any other session. No successful attack requires unwitting support from participants engaged in a different session.

Session separability clarifies the real world effects of a protocol. Suppose a protocol allows a customer to buy merchandise through a broker, who receives a commission from a manufacturer. Can the broker manipulate the protocol so one interaction with the customer allows two interactions with the manufacturer? Can the broker receive his commission for the same transaction twice?

Suppose a compliant customer and manufacturer interact with a dishonest broker in a separable protocol. If messages from the single customer run reach local runs M_1, M_2 for the manufacturer, they will belong to the same session. Alternatively one run M belongs to no session, i.e. it occurs with no involvement of a customer. These conditions are easy for a protocol designer to analyze. To protect against the first, the manufacturer should contribute a fresh random value ("nonce") to help define the session. Then distinct manufacturer runs always belong to different sessions. To protect against the second, some authentication is needed between manufacturer and customer, a familiar and well-understood problem. Thus, separability reduces the no-double-commission property to simple characteristics of the protocol.

Strand spaces offer a partially ordered model; protocol executions ("bundles") are annotated directed acyclic graphs [30,20,18]. The edges represent causal relations. We interpret separation properties in terms of the absence of causal paths in these graphs, or the ability to find a related graph without them.

Contributions. Our main result, the *Separability Theorem* (Thm. 18), tells how to take an execution of a protocol and modify it into another execution that satisfies separability. It applies to a range of different *separability specifications*. Each separability specification is a partial order that says which kinds of events are allowed to causally affect which others.

Our result about sessions, Thm. 19, says that the syntactic conditions in Def. 8—mainly concerning "session nonces" that serve to define sessions—entail the premise of Thm. 18. Thus, executions can be made to satisfy a separability specification defined in terms of these nonces. We also provide a transformation that strengthens any protocol to one that satisfies these conditions (Thm. 10).

Thm. 18 also applies to other separability specifications. We apply it to protocols with "disjoint encryption" in three slightly different senses (Thms. 21–24), thereby yielding variants, sometimes sharpenings, of a number of results on preserving security goals under protocol composition [19,1,9,8,11]. Thus, the Separability Theorem formalizes a pattern of reasoning with wide applicability in protocol design and analysis. It unites session-oriented reasoning with protocol composition.

Related work. Various flavors of sessions and separability have already played roles within protocol analysis and design. Among approaches based on computational methods, a session notion is often used to define the local runs that authentication should connect, as with Bellare and Rogaway [4]; in some models the sessions are defined by a bitstring that may be chosen by the adversary or built out of random contributions from the participants (e.g. Canetti-Krawczyk [7]). The Universal Composability model also assumes a random value that contributes to each cryptographically prepared unit and acts as a session identifier [6]; for a recent and more flexible alternative, see Küsters and Tuengerthal [24].

If different sessions of a protocol can never affect one another, then this simplifies analysis. The designer can explore the outcomes possible with a single instance of each role in the protocol. Indeed, Lowe's original proof that his changes to Needham-Schroeder were correct used a separability argument. He proved that any run could involve at most two non-separable instances of either role, and then model-checked the possible two-instance runs [25]. Lowe and Allaa Kamil [23] use separability to establish properties of TLS, such as that the adversary cannot divert application data from one TLS session to another. Their path-based methods within the strand space framework motivated some of the techniques used below in §4.

Cortier et al. [10] propose a protocol transformation, which they prove correct using session separability. Given a protocol satisfying any security property in an environment with a passive adversary, their transformation returns a new protocol that satisfies that property despite an active adversary. Their transformation adds freshly-generated nonces to the original protocol; this suggested our treatment of nonces in Thm. 10. Their transformation then inserts all of these nonces in with each message of the original protocol, which is signed and then encrypted. Our transformation does not add any additional cryptographic operations, but simply inserts the nonces into any pre-existing cryptographic units. This simpler treatment suffices because we are here exclusively concerned with separability, rather than any particular security goals. Another contrast concerns the adversary model. Their result concerns only sessions in which every participant is compliant, whereas our separability holds for the compliant participants, even in sessions with non-compliant participants.

Arapinis, Delaune, and Kremer [2] also offer a separability argument, leading to a protocol transformation which guarantees secrecy. The transformation adds nonces to each encrypted or signed term, although, together with nonces, it also adds principal names. Only the former is needed for separability; the latter helps with secrecy. Their transformation appears to generalize Lowe's fix to Needham-Schroeder. A subsequent paper with Ryan [12] investigates tagged password-based protocols, where the "tags" are tuples of session parameters hashed in with the key. They show that their composition is resistant to guessing attacks. Their proofs appear to establish particular instances of our Separability Theorem. We conjecture that our methods reconstruct their results, although proving this would require reformulating behavioral equivalences (in addition to trace properties) within our framework.

Deniélou et al. in [5] provide a compiler for generating ML code from multiparty protocol specifications. Their main result (Session Integrity Theorem) shows that there is no interference between multiple instances of the same protocol. Such a result could also be modeled in our framework as a variant of protocol independence. As we have mentioned, our approach also seems to capture the essence of several protocol composition results [19,1,9,8,11].

Separability also allows full verification within the bounded-session model of protocol analysis [29,26,3], rendering many classes of problems decidable.

In [17] we offer a logical language that can formalize the security goals that are preserved when omitting separable local runs.

2 Strand Spaces, with a Session Protocol

We first summarize the strand space terminology we will use in this paper. See [30,18] for more detail on strand spaces and our terminology.

We also introduce an example to illustrate separability, the *Trusted Broker Service*, in which a server S acts as a broker to match clients C_1 and C_2, who are executing different roles. S provides them with a key K to use to initiate an exchange. Clients trust the broker to generate a fresh key; to distribute it only to compliant principals; and to choose an appropriate pairing of clients.

Messages. Let Alg_0 be an algebra equipped with some operators and a set of homomorphisms $\eta\colon \mathsf{Alg}_0 \to \mathsf{Alg}_0$. We call the members of Alg_0 **basic values**.

Alg_0 is the disjoint union of infinite sets of *nonces*, *basic keys*, *names*, and *texts*. The operators $\mathsf{sk}(a)$ and $\mathsf{pk}(a)$ maps names to signature keys and public encryption keys. K^{-1} maps an asymmetric basic key to its inverse, and a symmetric basic key to itself. Homomorphisms η are maps that that respect sorts and the operators $\mathsf{sk}(a)$, $\mathsf{pk}(a)$, and K^{-1}. An infinite set X disjoint from Alg_0—the **indeterminates**—act like unsorted variables.

The algebra Alg of **messages** is freely generated from $\mathsf{Alg}_0 \cup X$ by two operations: encryption $\{\!|t_0|\!\}_{t_1}$ and tagged concatenation $\mathsf{tag}\ t_0\,,\ t_1$. The tags tag are drawn from some set TAG. For a distinguished tag nil, we write $\mathsf{nil}\ \ t_0\,,\ t_1$ as $t_0\,,\ t_1$. In $\{\!|t_0|\!\}_{t_1}$, a non-basic key t_1 is a symmetric key. To reduce cases in proofs, we do not introduce digital signature and hashing as separate operations. We can encode hashes $\mathsf{hash}(t)$ as encrypting t with a public key K_h, where no principal holds the inverse decryption key K_h^{-1}. A digital signature $[\![\,t\,]\!]_K$ is encoded as the concatenation $t\,,\ \{\!|\mathsf{hash}(t)|\!\}_K$.

A homomorphism $\alpha = (\eta, \chi)\colon \mathsf{Alg} \to \mathsf{Alg}$ pairs a homomorphism η on basic values and a function $\chi\colon X \to \mathsf{Alg}$; $\alpha(t)$ is defined by the conditions:

$$\begin{array}{llcr} \alpha(a) = \eta(a), & \text{if } a \in \mathsf{Alg}_0 & \qquad & \alpha(\{\!|t_0|\!\}_{t_1}) = \{\!|\alpha(t_0)|\!\}_{\alpha(t_1)} \\ \alpha(x) = \chi(x), & \text{if } x \in X & & \alpha(\mathsf{tag}\ t_0\,,\ t_1) = \mathsf{tag}\ \alpha(t_0)\,,\ \alpha(t_1) \end{array}$$

We call these homomorphisms **substitutions**, and use them to plug in values for parameters. Indeterminates x are blank slots, to be filled by any $\chi(x) \in \mathsf{Alg}$.

Messages t_1, t_2 have a *common instance* when there exist substitutions α, β that identify them: $\alpha(t_1) = \beta(t_2)$. Alg has the most general unifier property. That is, suppose that for $v, w \in \mathsf{Alg}$, there exist any α, β such that $\alpha(v) = \beta(w)$. Then there are α_0, β_0, such that $\alpha_0(v) = \beta_0(w)$, and for all α_1, β_1, if $\alpha_1(v) = \beta_1(w)$, then α_1 and β_1 are of the forms $\gamma \circ \alpha_0$ and $\gamma \circ \beta_0$.

Strands, Ingredients, and Origination. A **strand** is a sequence of local actions called **nodes**, each of which is either a message *transmission*, written $\bullet \rightarrow$, or else a message *reception*, written $\bullet \leftarrow$. Strands may be written vertically, or horizontally as in Fig. 1. This figure shows the behaviors of an initiating client C_1 and a responding client C_2 with a broker or server S. The protocol, which we call TBS, allows the broker to pair requests from suitable pairs of clients, and distribute a session key to them.

If n is a node, and the message t is transmitted or received, then we write $t = \mathsf{msg}(n)$. Double arrows indicate successive events on the same strand, e.g. $\bullet \Rightarrow \bullet \Rightarrow \bullet$. Each role in Fig. 1, and each local run in Figs. 5, 3, is a strand.

We write $s \downarrow i$ to mean the i^{th} node along s, starting at $s \downarrow 1$. The **parameters** of s are the basic values and indeterminates in any $\mathsf{msg}(s \downarrow i)$.

The **ingredients** of a message are those subterms that may be reached by descending through concatenations, and through the plaintext but not the encryption keys. The values that **occur** in it descend also through encryptions. We write $\sqsubseteq$ ("is an ingredient of") and $\ll$ ("occurs in"), resp., for the smallest reflexive, transitive relation such that

$$t_1 \sqsubseteq (t_1\,,\, t_2) \qquad t_2 \sqsubseteq (t_1\,,\, t_2) \qquad t_1 \sqsubseteq \{\!|t_1|\!\}_{t_2}$$
$$t_1 \ll (t_1\,,\, t_2) \qquad t_2 \ll (t_1\,,\, t_2) \qquad t_1 \ll \{\!|t_1|\!\}_{t_2} \qquad t_2 \ll \{\!|t_1|\!\}_{t_2}.$$

We say that t **originates** on a node n if n is a transmission node, and $t \sqsubseteq \mathsf{msg}(n)$, and for all n_0, if $n_0 \Rightarrow^+ n$, then $t \not\sqsubseteq \mathsf{msg}(n)$. A basic value a is *freshly chosen* if it originates just once. We call it **uniquely originating**. In this case, a was chosen by a participant, without the bad luck of any other principal selecting the same value independently. A key is regarded as uncompromised if it originates *nowhere*. We call a basic value a **non-originating** in B if there exists no node $n \in B$ such that a originates at n. It may still be used in B even if it does not originate anywhere, since the regular strands may receive and send messages encrypted by K or K^{-1}, thus using K for encryption and decryption, resp.

A message t_0 **lies only inside encryptions in** t **with keys** S iff, in t's abstract syntax tree, every path from the root to an occurrence of t_0 traverses an encryption, and if that occurrence is in the plaintext, then the key is in S.

Protocols. A **protocol** Π is a finite set of strands, called the **roles** of the protocol. A **regular** strand for Π is any instance of one of the roles of Π, i.e. the result $\alpha(\rho)$ of some substitution α on the parameters of a role $\rho \in \Pi$. Fig. 1 is an example protocol. A protocol may also specify some parameters of a role that are always non-originating or uniquely originating. We will also formalize *adversary* behavior by strands (which use inverse). We stipulate a syntactic constraint:

Assumption 1 *If $\rho \in \Pi$, then the key inverse symbol does not appear in any message $\mathsf{msg}(\rho \downarrow i)$. Moreover, $\mathsf{sk}(A) \not\sqsubseteq \mathsf{msg}(\rho \downarrow i)$. If $\{\!|t|\!\}_K \ll \mathsf{msg}(\rho \downarrow i)$ for $\rho \in \Pi$, then K is either a basic value or an encryption (not a concatenation).*

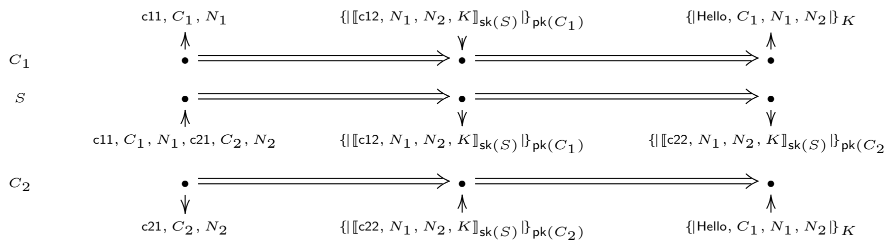

Fig. 1. Trusted Broker Service Protocol, TBS

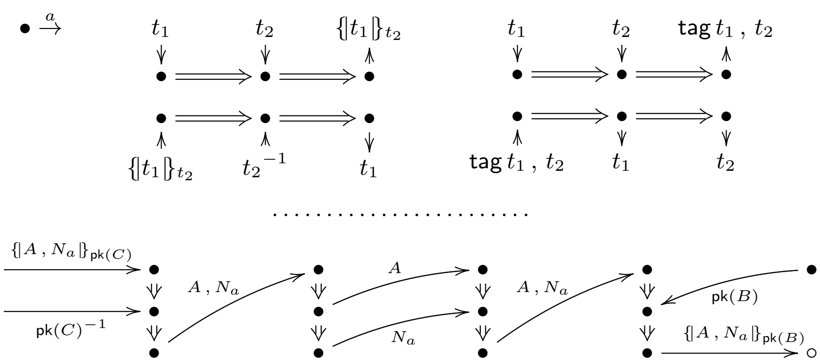

Fig. 2. Part I: Adversary roles to generate basic value a; encrypt and decrypt; concatenate and separate. Part II: A compound adversary activity

The Adversary. Adversary strands consist of zero or more reception nodes followed by one transmission node. The adversary obtains the transmitted value as a function of the values received; or creates it, if there are no reception nodes. The adversary can choose basic values, and operate on complex values using the strands shown above in Fig. 2. These are often used in patterns, e.g. as in Fig. 2 Part II, which transport information along paths. Six strands are shown. Two are of length 1, in which the adversary transmits keys, namely his own private decryption key $\mathsf{pk}(C)^{-1}$ and B's public encryption key $\mathsf{pk}(B)$. Two are a (leftmost) decryption strand and a (rightmost) encryption strand. The second node on a decryption or encryption strand is called the **key node**, since it receives the key used to perform the cryptographic operation.

In the middle are a separation strand that breaks A, N_a into its two parts, followed by a concatenation strand that puts them back together. These strands are unnecessary here. We include them here to illustrate that the adversary can always break a concatenation down to non-concatenated parts, i.e., either basic values or encryptions (see Assumption 3).

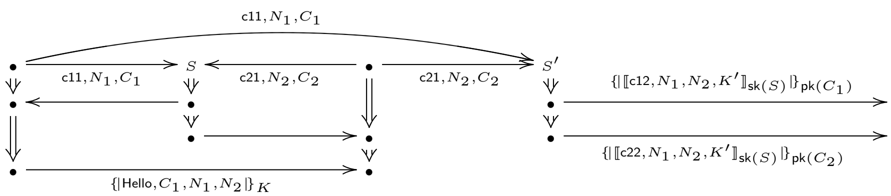

Fig. 3. A bundle of protocol TBS

Adversary strands are closed under substitutions along the strand, as they comprise all the instances of the roles in Fig. 2, Part I. Indeed, this also holds for regular strands, which are all the substitution instances of the roles $\rho \in \Pi$:

Lemma 1. *If α is a substitution and s is an adversary strand or a regular strand of Π, then so is $\alpha(s)$.*

Bundles. An execution is pieced together from a finite set of strands (or their initial segments), where these may be regular strands of Π or adversary strands from §4. Two nodes are connected with a single arrow $\bullet \to \bullet$ when the former transmits a message, and the latter receives that same message directly from it. A *bundle* is a causally well founded graph built using strands by $\to$:

Definition 2. *Let $\mathcal{B} = \langle \mathcal{N}, \to_E \cup \Rightarrow_E \rangle$ be a finite, directed acyclic graph where (i) $n_1 \Rightarrow_E n_2$ implies $n_1 \Rightarrow n_2$, i.e. that n_1, n_2 are successive nodes on the same strand; and (ii) $n_1 \to_E n_2$ implies that n_1 is a transmission node, n_2 is a reception node, and $\mathsf{msg}(n_1) = \mathsf{msg}(n_2)$. $\mathcal{B}$ is a **bundle** if:*

1. *If $n_1 \Rightarrow n_2$, and $n_2 \in \mathcal{N}$, then $n_1 \in \mathcal{N}$ and $n_1 \Rightarrow_E n_2$; and*
2. *If n_2 is a reception node, there exists a unique $n_1 \in \mathcal{N}$ such that $n_1 \to_E n_2$.*

*$\mathcal{B}$ is an **open bundle** if, in condition 2, there is* at most one *$n_1 \in \mathcal{N}$ such that $n_1 \to_E n_2$, rather than exactly one.*

We write $\mathsf{nodes}(\mathcal{B})$ for the nodes of $\mathcal{B}$, and $\mathsf{regnodes}(\mathcal{B})$ for its regular (non-adversary) nodes; $\mathsf{edges}(\mathcal{B})$ is the set $\Rightarrow_E \cup \to_E$ of edges of $\mathcal{B}$. $\preceq_{\mathcal{B}}$ is the causal partial order $(\to_E \cup \Rightarrow_E)^*$, and $\prec_{\mathcal{B}} = (\to_E \cup \Rightarrow_E)^+$.

*A node n is **realized** in an open bundle $\mathcal{B}$ iff n is a transmission node, or else n is a reception node and has an incoming $\to$ edge, i.e. $n' \to n$.*

*$(\mathcal{B}, \mathsf{unique}, \mathsf{non})$ is an **annotated** bundle (resp. open bundle) if $\mathcal{B}$ is a bundle (resp. open bundle), unique is a finite set of basic values each originating at most once in $\mathcal{B}$, and non is a finite set of basic values each originating nowhere in $\mathcal{B}$.*

The causal partial order $\preceq$ is well-founded, since $\mathcal{B}$ is finite.

Fig. 3 is a bundle. TBS defines a session via a nonce from each client, and the server-generated session key. It gathers the two incoming messages to the broker in a single reception, allowing some (untrusted) auxiliary process to propose a matching. In Fig. 3, the adversary reuses the nonces N_1, N_2 to start a second server strand. However, we can fix this, separating the second server strand, just by renaming these nonces to new values N_1', N_2'. This yields the new bundle in Fig. 4, in which the adversary can supply the message coming from the upper right. Fig. 4 is an open bundle, as shown without adversary activity.

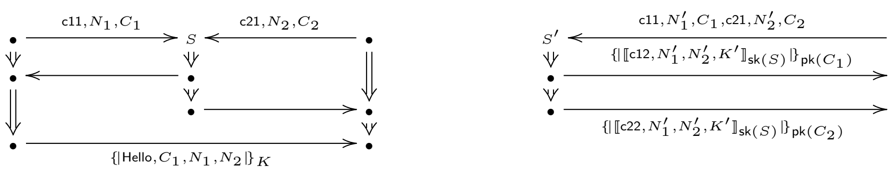

Fig. 4. Open bundle separating Fig. 3

These figures are annotated (possibly open) bundles with various choices of $\mathsf{unique}, \mathsf{non}$. An interesting choice would be $\mathsf{unique} = \{N_1, N_2, K, N_1', N_2'\}$ for Fig. 4 and $\mathsf{unique} = \{N_1, N_2, K\}$ for Fig. 3. A relevant choice for both $\mathsf{non} = \{\mathsf{sk}(S), \mathsf{pk}(C_1)^{-1}, \mathsf{pk}(C_2)^{-1}, \mathsf{pk}(S)^{-1}\}$.

In studying separability we are interested in bundles equipped with a choice of fresh and uncompromised values. Hence, we will assume that all bundles are annotated with sets of uniquely originating and non-originating values $\mathsf{unique}, \mathsf{non}$. When using "bundle" and $\mathcal{B}$, we will mean "annotated bundle" as defined above.

The core pattern for separating a session is:

- removing dependence on an existing session;
- renaming some freshly chosen items in one or more local runs;
- allowing the adversary to supply incoming messages in these runs.

When a protocol ensures that this pattern will succeed in separating behaviors, it has session behavior.

However, this is not always possible. As an example, consider the protocol TBSMINUS, which is just like TBS, except that the session nonces N_1, N_2 are omitted in all the messages. Here we can have the essentially inseparable bundle Fig. 5. No amount of renaming and pruning edges will produce a bundle in which C_2 and C_2' do not both depend on the same strand C_1.

We assume (i) public encryption keys may be freely sent or used by anyone, including the adversary; and (ii) when a value a originates uniquely, and is used on a different regular strand as part of a key for encryption, then it has been received as an ingredient on that strand. When $a \sqsubseteq \mathsf{msg}(m_1)$, this conclusion follows from the definition of unique origination. We also assume (iii) that a basic value is not received from a later transmission when it could be received from an earlier one. If a bundle violates this property, we can fix it by rerouting arrows to start from earlier nodes.

Assumption 2 *Let* $(\mathcal{B}, \mathsf{unique}, \mathsf{non})$ *be an annotated bundle.*

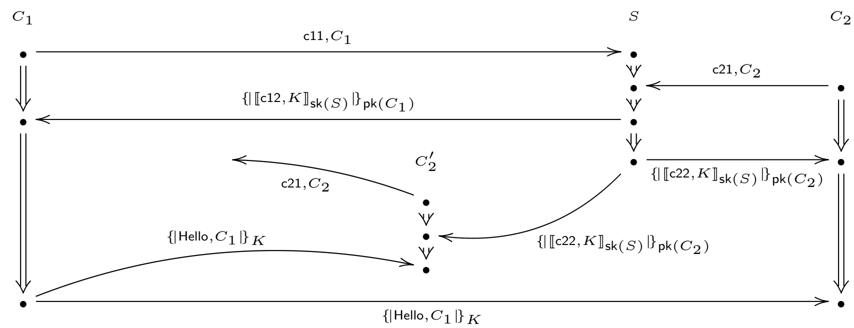

Fig. 5. An inseparable execution of TBSMINUS

1. $\mathsf{pk}(A) \notin \mathsf{unique} \cup \mathsf{non}$ *for all names* A.
2. *Suppose* $a \in \mathsf{unique}$*,* a *originates on* n_0*, and for some transmission node* m_1*,* $a \ll K$ *and* $\{\!|t|\!\}_K$ *originates at* m_1*. If* n_0, m_1 *lie on different strands, then there is a reception node* $m_0 \Rightarrow^+ m_1$ *such that* $a \sqsubseteq \mathsf{msg}(m_0)$*.*
3. *If* $a = \mathsf{msg}(n_0) = \mathsf{msg}(n_1) = \mathsf{msg}(n_2)$ *is a basic value, where* n_0, n_1 *are transmission nodes, with* $n_0 \preceq n_1 \preceq n_2$*. Then it is not the case that* $n_1 \to n_2$*.*

Lemma 3. *Suppose* $\mathcal{B}$ *is a bundle with* $n_0, m_1 \in \mathsf{nodes}(\mathcal{B})$*. If* $a \in \mathsf{unique}$ *originates at* n_0 *and* $a \ll \mathsf{msg}(m_1)$*, then* $n_0 \preceq m_1$*.*

Proof. If there are any counterexamples m_1, choose m to be $\preceq$-minimal among them. By clause 2 in the definition of bundle, m is not a reception node. If $a \sqsubseteq \mathsf{msg}(m_1)$, then m is a point of origination, so by uniqueness $m = n_0$. Otherwise, Assumption 2, clause 2 contradicts the minimality of m. □

The "Lies-below" relation. We now define a relation between bundles (or open bundles) of reducing information. We say that one (open) bundle lies below another when the latter results by adding information to the ordering relation $\preceq$ and adding equations between parameters. The key idea is reducing the ordering relation in a bundle $\mathcal{B}$, possibly renaming some occurrences of parameters, so as to "rename them apart" in a simpler bundle $\mathcal{C}$. We actually formalize this in the other direction, by considering a homomorphism α from $\mathcal{C}$ into the richer $\mathcal{B}$. We call this a *local renaming*, because restricted to portions of $\mathcal{C}$ it acts like a renaming. It acts injectively on each portion separately.

Definition 4 (Local Renaming). *Suppose* $\mathcal{C}$ *is an open bundle.*

The sets $S_1, \ldots, S_n$ ***partition*** $\mathsf{nodes}(\mathcal{C})$ ***by strands*** *if (i) the* S_i *are disjoint; (ii)* $\bigcup S_i = \mathsf{nodes}(\mathcal{C})$*; and (iii) any two nodes on the same strand are in the same partition class* S_i*.*

A substitution α *is a* ***local renaming of*** $\mathcal{C}$ with respect to $S_1, \ldots, S_n$ *if the sets partition* $\mathsf{nodes}(\mathcal{C})$ *by strands, and moreover, for every* $j \leq n$*,* α *restricted* *to the parameters of the strands in S_j is a renaming, i.e. an invertible map from parameters to parameters.*

For instance, in Fig. 4, the part to the left of the white space S_1 and the part to the right S_2 form a partition by strands. The map which sends $N'_1 \mapsto N_1$ and $N'_2 \mapsto N_2$, and is elsewhere the identity, is a local renaming, which we will write $[N'_1 \mapsto N_1, N'_2 \mapsto N_2]$. It is a renaming (the identity) when restricted to the parameters that appear in S_1, the left half, since N'_1, N'_2 do not appear on the left. Moreover, it is a renaming when restricted to S_2, the right half, too, since N_1, N_2 do not appear as parameters on the right. Thus, it is injective on the parameters appearing in S_2.

Every renaming is a local renaming, but a local renaming α is not a true ("global") renaming when $\alpha(x) = \alpha(y)$ holds for parameters x, y to nodes in different partition classes S_j, S_k. We often think of the action of a local renaming backward, viewing its source as the result of "renaming apart" values that are equated in its target. If we view $[N'_1 \mapsto N_1, N'_2 \mapsto N_2]$ as if it were acting on Fig. 3 to yield Fig. 4, then it is "renaming apart" different occurrences of N_1, N_2.

One open bundle lies below another if, after applying a local renaming forward, their regular nodes are the same, as are their uniquely originating and non-originating values, but one precedence order is a suborder of the other:

Definition 5. *1. $\mathcal{C}$ **lies below** $\mathcal{B}$ **via** α iff, for some $S_1, \ldots, S_i$, α is a local renaming for $\mathcal{C}$ with respect to $S_1, \ldots, S_i$, and:*

(a) $\alpha(\mathsf{regnodes}(\mathcal{C})) = \mathsf{regnodes}(\mathcal{B})$;

(b) For all $n_0, n_1 \in \mathsf{regnodes}(\mathcal{C})$, $n_0 \preceq_{\mathcal{C}} n_1$ *implies* $\alpha(n_0) \preceq_{\mathcal{B}} \alpha(n_1)$;

(c) $\alpha^{-1}(\mathsf{unique}(\mathcal{B})) = \mathsf{unique}(\mathcal{C})$; *and*

(d) $\alpha^{-1}(\mathsf{non}(\mathcal{B})) = \mathsf{non}(\mathcal{C})$

*2. $\mathcal{C}$ **lies below** $\mathcal{B}$ if it does so via some α.*

*3. $\mathcal{B}$ and $\mathcal{C}$ are **equivalent** iff each lies below the other via renamings α, β, and $\alpha \circ \beta$ is the identity on the parameters involved.*

For instance, Fig. 4 lies below Fig. 3 via $[N'_1 \mapsto N_1, N'_2 \mapsto N_2]$, given the choices of $\mathsf{unique}, \mathsf{non}$ mentioned after Defn. 2.

If $\mathcal{C}$ lies below $\mathcal{B}$, then $\mathcal{C}$ differs from $\mathcal{B}$ only in having a sparser ordering, and in not yet having equated some parameters that have been equated in $\mathcal{B}$. We can think of $\mathcal{C}$ as a simplified version of $\mathcal{B}$. It is less informative in that the information that these parameters are equal has not yet been added.

Lemma 6. *"Lies below" is a well-founded partial order to within isomorphism:*

1. *"Lies below" is reflexive and transitive.*
2. *$\mathcal{C}$ and $\mathcal{B}$ each lie below the other iff their regular parts are isomorphic.*
3. *If $\langle \mathcal{B}_i \rangle_i$ is an infinite sequence of bundles such that $i < j$ implies $\mathcal{B}_j$ lies below $\mathcal{B}_i$, then for some i, k, $i < k$ and $\mathcal{B}_i$ lies below $\mathcal{B}_k$.*

3 Formalizing Sessions

We now turn to defining when TBS and similar session-oriented protocols are separable. Suppose that Π is a protocol, and $P\colon \Pi \to \mathsf{Nonce} \cup \mathsf{Key}$ is a function that chooses a parameter for each role. As an example, if Π is TBS, we would be interested in the function P that assigns N_1 to the first client role; N_2 to the second client role; and K to the server (broker) role.

We say that x is a **session parameter** if $x \in \mathsf{range}(P)$. P associates each role to the session parameter that it chooses. We call $P(\rho)$ ρ's **proper session parameter**, and we require that $P(\rho)$ originates on ρ.

If x is a session parameter, x **is acquired at step** i if $x \ll \rho \downarrow i$ but $x \not\ll \rho \downarrow j$ for $j < i$. It is **acquired by step** k if it is acquired at step i for some $i \leq k$. A parameter x is **key material at step** i if $x \ll K$ and $\{|t|\}_K \ll \mathsf{msg}(\rho \downarrow i)$.

As a convention, we will assume that the parameters of each role have been chosen (by a renaming if necessary) so that corresponding session parameters on different roles have the same name. We could of course avoid this convention at the cost of added notation, in the form of a function which would supply the necessary correlations.

No Ambiguity. TBS uses the session parameters unambiguously in each encryption. No encryption in the protocol could be misinterpreted by a receiver so as to interchange the session parameters. For instance, N_1 and N_2 always appear in the same order, and K always appears after them or in key position.

No ambiguity: If encryptions $\{|t|\}_K \ll \mathsf{msg}(\rho \downarrow i)$ and $\{|t'|\}_{K'} \ll \mathsf{msg}(\sigma \downarrow j)$ have a common instance $\alpha(\{|t|\}_K) = \beta(\{|t'|\}_{K'})$, then ρ and σ have acquired the same session parameters by steps i and j resp., and $\alpha(x) = \beta(x)$ for each of those session parameters.

We here follow our convention that corresponding session parameters on different roles have been given the same parameters names.

Contribution. Every encrypted unit involves the session parameters. This is akin to the tagging property [12], except that the session parameters do not have to contribute to the key. The last message of TBS is $\{|\mathsf{Hello}, C_1, N_1, N_2|\}_K$. Two session parameters are in the plaintext, while K is the encryption key. All the session parameters could be concentrated in the key; $\{|\mathsf{Hello}, C_1|\}_{\mathsf{hash}(N_1,N_2,K)}$ would also work. Alternatively, they could all be concentrated in the plaintext, with some public key used for encryption.

In this protocol, the participants agree on all of the session parameters at the start. They then use them throughout the remainder of the protocol. A protocol can also have some participants agree on their session parameters, while other participants join the session later. These "late arrivals" allow for an attractive flexibility in the session-type literature [13]. Of course, the encrypted units *before* the late arrivals are expected to contain only the session parameters that have already been seen at that point.

Contribution to encryptions: If $\{|t|\}_K \ll \mathsf{msg}(\rho \downarrow i)$ and session parameter x is acquired by step i, then $x \ll \{|t|\}_K$.

The No-Vs property. The observation that session parameters may be acquired piecemeal is an important insight. It implies that "same session," which sounds like an equivalence relation, is in fact misleading. A partially defined session with session parameters $x_1, \ldots, x_i$ may affect any of its possible extensions with an additional session parameter $x_1, \ldots, x_i, x_{i+1}$. However, any one of those extensions is incompatible with those having a different value x'_{i+1}. Indeed, messages from a step with extended session parameters $x_1, \ldots, x_i, x_{i+1}$ should not affect an event with session parameters $x_1, \ldots, x_i$. If they did, the latter could also affect a distinct extension $x_1, \ldots, x_i, x'_{i+1}$. Thus, transitively, there could be effects from an event with parameters $x_1, \ldots, x_i, x_{i+1}$ to one with parameters $x_1, \ldots, x_i, x'_{i+1}$. That would be contrary to the session discipline.

For this reason, we regard the "may influence" relation on partially defined sessions as a partial order (on the sessions) or as a pre-order (on the transmission and reception events within the sessions). We will write $n_1 \rightsquigarrow n_2$ when an event n_1 may influence an event n_2.

We require non-influence to persist, specifically when n_1 selects a fresh value that is a parameter to n_2. We formulate this as a "no Vs" condition. Whenever we have a V in the may-influence relation, this is not an open V, but a closed triangle-like configuration, for any $n_3 \succeq n_2$:

$$\begin{tikzcd}[row sep=small] n_1 \arrow[dr, squiggly] & \\ & n_3 \\ n_2 \arrow[ur, squiggly] & \end{tikzcd} \quad \text{implies} \quad \begin{tikzcd}[row sep=small] n_1 \arrow[dd, squiggly] \arrow[dr, squiggly] & \\ & n_3. \\ n_2 \arrow[ur, squiggly] & \end{tikzcd} \tag{1}$$

A node n_2 that I cannot influence cannot influence a later node that I can influence, at least when I have uniquely originated a value found in that node. This no Vs property turns out to be crucial to proving the Separability Theorem, whose proof tries to create new bundles by local renamings.

To see what could go wrong, suppose the TBS server received the two parts of its first incoming message on separate nodes: $(\mathsf{c11}, C_1, N_1) \Rightarrow (\mathsf{c21}, C_2, N_2) \Rightarrow \ldots$. Then an adversary could deliver C_2's nonce N_2 as if it were from C_1, on the first server node n_1. C_2's first node n_1 should not influence n_2, since n_1 has the C_2 nonce defined, whereas n_2 does not; n_2 has only the C_1 nonce parameter defined. However, if the adversary re-delivers the same nonce on the server's second node n_3, then C_2's first node n_1 can influence this second server node n_3. Node n_3 has the same value for the only session parameter defined on n_1. This is precisely the open V situation, where $n_1 \not\rightsquigarrow n_2 \rightsquigarrow n_3$, and $n_1 \rightsquigarrow n_3$.

Acquisition. In order to ensure the no-Vs property syntactically, some properties are needed, constraining how session parameters are acquired. First, some session parameters $\overline{x}$ are received in a principal's first reception. These may be transported without encryption, such as N_1 and N_2 in TBS. This is why S receives both N_1 and N_2 in a single message in its first node. Second, there are no transmissions after a reception and before transmitting a strand's proper session nonce. Third, when a session includes late-arriving participants, values freshly chosen after a late arrival in the session will be transmitted under encryptions that cannot be compromised. Various techniques are available for proving this [20,18], but here we will just use a simple sufficient condition, namely that the decryption key is non-originating. These messages will be received by participants that have already joined the session; i.e. their proper nonces have already been chosen, and must also appear in this encryption by the *Contribution* requirement. This is a per-bundle requirement, for a bundle $\mathcal{B}$.

Parameter acquisition: Session parameters divide into two groups, $\overline{x}$ and $\overline{y}$.

1. If x in $\overline{x}$ is acquired on reception node $\rho \downarrow i$, then i is the earliest reception node on ρ.
2. If x in $\overline{x}$ is acquired on transmission node $\rho \downarrow i$, and $\rho \downarrow k$ is any reception node with $k < i$, then there is no transmission node between them.
3. Let y in $\overline{y}$ be acquired (by reception or transmission) on $\rho \downarrow i$, and let $k \geq i$. There is a set $\mathsf{LAK}(\mathcal{B})$ of *late-arrival protection keys of* $\mathcal{B}$ such that: (a) If $\alpha(\rho \downarrow k) \in \mathsf{nodes}(\mathcal{B})$, then $\alpha(y)$ lies only inside encryptions in $\mathsf{msg}(\alpha(\rho \downarrow k))$ with keys K where $\alpha(K^{-1}) \in \mathsf{LAK}(\mathcal{B})$.
   (b) If $a \in \mathsf{unique}_{\mathcal{B}}$ is any value acquired on $\alpha(\rho \downarrow k)$, $\alpha(y)$ lies only inside encryptions in $\mathsf{msg}(\alpha(\rho \downarrow k))$ with keys K where $\alpha(K^{-1}) \in \mathsf{LAK}(\mathcal{B})$.

Condition 3 ensures that y always appears together with all previously defined session parameters. We focus on bundles in which, for any late arrivals to the session in a bundle $\mathcal{B}$, the strands still active then are all uncompromised, i.e. $\mathsf{LAK}(\mathcal{B}) \subseteq \mathsf{non}_{\mathcal{B}}$. In TBS, all session parameters belong to the first group $\overline{x}$, as all of the roles acquire them from their peers on their first reception. For protocol design, it is desirable that the session key can double as S's session parameter, traveling in the encrypted messages from the server.

May-influence relations. Curiously, Thm. 18 depended only on two properties of a reflexive, transitive *may-influence* relation, namely, the no Vs property, and the fact that forward influence on a strand is always permitted. Because of this generality, we sought to specify various degrees of separability, i.e. to specify how sparse a bundle we would like to obtain in the "lies below" ordering. To parametrize our reasoning, we define a *may-influence* relation to be a preordering $n_1 \rightsquigarrow n_2$ on regular nodes with these two properties. It specifies the upper bound on the set of Π nodes allowed to influence other Π nodes.

Definition 7. *Let $\mathcal{B}$ be an (annotated) bundle for protocol Π. Then a preorder $\rightsquigarrow$ is a **may-influence relation** for $\mathcal{B}$ iff for all $n_1, n_2, n_3 \in \mathsf{regnodes}(\mathcal{B})$,*

1. *if $n_1 \Rightarrow n_2$ then $n_1 \rightsquigarrow n_2$; and*
2. *"No Vs," Eqn. 1: Suppose (i) $a \in \mathsf{unique}_{\mathcal{B}}$ originates at n_1 and $a \ll \mathsf{msg}(n_2)$ and (ii) $n_2 \rightsquigarrow n_3$ and $n_2 \preceq_{\mathcal{B}} n_3$. If $n_1 \rightsquigarrow n_3$, then $n_1 \rightsquigarrow n_2$.*

*$\mathcal{B}$ **obeys** $\rightsquigarrow$ iff, for all $m, n \in \mathsf{regnodes}(\mathcal{B})$, $m \preceq_{\mathcal{B}} n$ implies $m \rightsquigarrow n$.*

*Π **obeys** $\rightsquigarrow$ subject to Φ if, for every Π-bundle $\mathcal{B}$ satisfying Φ, there is a Π-bundle $\mathcal{C}$ satisfying Φ such that $\mathcal{C}$ lies below $\mathcal{B}$ and $\mathcal{C}$ obeys $\rightsquigarrow$.*

When $m \rightsquigarrow n$, we say that m is *permitted to influence* n.

When $m \Rightarrow n$, m must be allowed to influence n, since it is impossible to prevent the influence; hence condition 1 on influence functions. Condition 2 prohibits open, V-shaped configurations. One leg of the V starts at a's origin n_1, and the other at n_2, and the legs join at a jointly influenced n_3. When $a \ll \mathsf{msg}(n_2)$, then n_1 must be permitted to influence n_2. If a's origin n_1 cannot influence n_2, then their causal consequences must remain separated thereafter.

Π obeys $\rightsquigarrow$ if Π-bundles either already obey the ordering constraint, or some bundle lying below is sparse enough to obey it. In weakening the order $\preceq$, we are allowed to select preimages under local renamings. We use the constraints Φ to record assumptions about freshly chosen nonces and uncompromised keys.

Protocols with session parameters. We can now define:

Definition 8. *A bundle $\mathcal{B}$ satisfies Φ_s, the* ***session constraint****, if the late arrival protection keys $\mathsf{LAK}(\mathcal{B}) \subseteq \mathsf{non}_{\mathcal{B}}$ and, for every node $\alpha(\rho \downarrow i) \in \mathsf{nodes}(\mathcal{B})$, where ρ acquires its proper session nonce at step i, $\alpha(P(\rho)) \in \mathsf{unique}_{\mathcal{B}}$.*

Π has ***session parameters*** *P for $\mathcal{B}$ if* No ambiguity*,* Contribution to encryptions*, and* Parameter acquisition *hold for Π, P, and $\mathcal{B}$.*

The ***session may-influence*** *relation $\rightsquigarrow_{\mathsf{s}}$ holds between Π-nodes n_1 and n_2, written $n_1 \rightsquigarrow_{\mathsf{s}} n_2$, iff (i) $n_1 = \alpha(\rho \downarrow i)$ and $n_2 = \beta(\sigma \downarrow j)$ where $\rho, \sigma \in \Pi$; (ii) every session parameter x that has been acquired by step i on ρ has been acquired by step j on σ; and (iii) $\alpha(x) = \beta(x)$ for each session parameter x acquired by step i on ρ.*

Essentially, $n_1 \rightsquigarrow_{\mathsf{s}} n_2$ means that the partial function assigning session parameters to values in node n_1 is a subfunction of the partial function assigning session parameters to values in node n_2. The may-influence relation is fixed by the ordering of definedness on these partial functions.

Lemma 9. *If $\mathcal{B}$ is a Π-bundle satisfying Φ_s, and Π has session parameters P in $\mathcal{B}$, then $\rightsquigarrow_{\mathsf{s}}$ is a may-influence relation for $\mathcal{B}$.*

Proof. Condition 1, that $\rightsquigarrow_{\mathsf{s}}$ is preserved by $\Rightarrow$, is immediate from the fact that session parameters are unchanged once defined.

For the no-Vs property 2, suppose that a originates uniquely on $n_1 = \alpha(\rho \downarrow i)$, and is received on $n_2 = \beta(\sigma \downarrow j)$. Suppose first that i is before any $\overline{y}$ parameter originates or is acquired. By the *Parameter acquisition* condition 1, all of the $\overline{x}$ parameters except possibly $P(\sigma)$ are determined on n_2. If $n_1 \not\rightsquigarrow_{\mathsf{s}} n_2$, then either n_2 disagrees with n_1 on a session parameter, or n_2 has not yet acquired a x acquired on n_1.

If the former, this disagreement persists to any $n_3 \succeq n_2$ such that $n_2 \rightsquigarrow_{\mathsf{s}} n_3$.

If the latter, it can only be $P(\sigma)$. Since $\mathcal{B}$ satisfies Φ_s, when the strand of n_2 acquires a value for its proper session parameter, it will be uniquely originating, hence distinct from the value previously defined on n_1. Moreover, by *Parameter acquisition* condition 2, $P(\sigma)$ is chosen by the time of the next transmission after n_2. Hence, any subsequent node that n_2 can influence has a value for $P(\sigma)$ incompatible with n_1's.

Finally, suppose that i is at or after the acquisition of some late-arriving y session parameter. Then a is transmitted inside a safe encryption, and any regular strand that receives it must retransmit it safely. Hence, a is accompanied by all session parameters in force at that point, inside the same encrypted unit (*Contribution to encryptions*). Moreover, any recipient must agree on the values of those parameters, by *No ambiguity*. Thus $n_1 \rightsquigarrow_{\mathsf{s}} n_2$, which implies No-$V$s. $\square$

A transformation yielding protocols with session parameters. Theorem 20 suggests a transformation to produce protocols with session parameters.

The transformation has two parts. The first part prepends before σ a node that transmits a session parameter, and a node that receives a concatenated tuple containing session nonces from each of the other roles:

$$+N_i \;\Rightarrow\; -(N_1\,,\,\ldots\,,\,N_{i-1}\,,\,N_{i+1}\,,\,\ldots\,,\,N_k) \;\Rightarrow\; \sigma$$

In the second part, we transform all encrypted units $\{\!|t|\!\}_K$ contained in σ, to $\{\!|t\,,\,\tilde{N}|\!\}_K$, where $\tilde{N}$ is the sequence of all the session nonces introduced in the first step. Thus, letting $\mathcal{T}_{\tilde{N}}$ be this transformation,

Theorem 10. *$\mathcal{T}_{\tilde{N}}(\Pi)$ has session parameters for each $\mathcal{T}_{\tilde{N}}(\Pi)$-bundle $\mathcal{B}$.*

It is easy to very that *No Ambiguity*, *Contribution to Encryptions*, and *Parameter Acquisition* are all true, where the late-arriving parameters $\overline{y}$ are vacuous.

4 The Separability Theorem

Penetrator paths. The ways that adversary strands manipulate messages are tightly constrained by their syntactic forms. We introduce *penetrator paths* to be able to express these relations conveniently.

Definition 11. *A **key node** is the middle node on an adversary encryption or decryption strand, which receives the key to be used (Fig. 2).*

*A **penetrator path** in $\mathcal{B}$ is a sequence $p = \langle n_0, n_1, \ldots, n_k\rangle$ with $k > 0$ and each $n_i \in \mathsf{nodes}(\mathcal{B})$, such that:*

1. *$n_1, \ldots, n_{k-1}$ are all penetrator nodes;*
2. *if n_i is a reception node and $i < k$, then n_{i+1} is a transmission node and $n_i \Rightarrow^+ n_{i+1}$;*
3. *if n_i is a transmission node, then $n_i \rightarrow n_{i+1}$ in $\mathcal{B}$.*

We often focus on the penetrator paths that stretch from a regular node to a regular node, traversing penetrator strands. These represent activities of the adversary that extract useful materials from regular transmissions, and use them to construct messages to satisfy regular receptions.

We write $p(i)$ for the node n_i, and $|p|$ for k, the number of arrows traversed by p, so $p(|p|)$ is the last node on p. Two paths are shown in Fig. 2. In both cases, $p(0)$ is the hollow circle at the upper left, indicating an unshown regular node, and $p(9)$ is the hollow circle at lower right. One path traverses the edge A in the middle, and the other traverses N_a. We generally write $\mathsf{first}(p)$ and $\mathsf{last}(p)$ for $p(0)$ and $p(|p|)$.

Definition 12. *The penetrator path p is **direct** if no key node appears in p, except possibly as* $\mathsf{last}(p)$*.*

*$\mathcal{B}$ is **normal** if, on every direct penetrator path, each destructive penetrator strand (decryption, separation) appears before any constructive strand (encryption, concatenation).*

The penetrator paths in Fig. 2 are direct. We speak of an *extended path* when we wish to emphasize that it may not be direct.

Lemma 13 ([20]). *Every bundle $\mathcal{B}$ has a normal bundle $\mathcal{C}$ lying below $\mathcal{B}$ via the identity* Id*. If $\mathcal{C}$ is any normal bundle, and p is a direct penetrator path in $\mathcal{C}$, then there is a pair of nodes $p_j \to p_{j+1}$ such that, for all $i \leq j \leq k$:*

1. $\mathsf{msg}(p(i)) \sqsubseteq \mathsf{msg}(\mathsf{first}(p))$ *and* $\mathsf{msg}(p(k)) \sqsubseteq \mathsf{msg}(\mathsf{last}(p))$*;*
2. *If $p(i)$ is an adversary node, then $p(i)$ lies on a destructive strand (decryption, separation); and*
3. *If $p(k+1)$ is an adversary node, then $p(k+1)$ lies on a constructive strand (encryption, concatenation).*

This lemma still holds in our current context, which includes compound keys, because it is restricted to *direct* paths p. Since a key node in p must be the last node, and we never continue along its encryption or decryption strand, we never encounter any case different from those already shown in the proof in [20].

By this lemma, when proving that there exists a bundle lying below $\mathcal{B}$ with a particular property, it is sound to silently assume that $\mathcal{B}$ is normal.

The **bridge term** of a direct penetrator path p in a normal $\mathcal{B}$ is the message $\mathsf{msg}(p(j))$ on the edge that follows all destructive penetrator strands and precedes all constructive penetrator strands. We will write $\mathsf{bt}(p)$ to refer to the bridge term of p. A single communication edge $\mathsf{first}(p) \to p(1)$, with no adversary strands in between, is a direct path of length 1; $\mathsf{bt}(p) = \mathsf{msg}(\mathsf{first}(p)) = \mathsf{msg}(p(1))$. The two edges leading to n_1 and n_2 in Fig. 3 are examples with the concatenated bridge terms $\mathsf{c11}, N_1, C_1$ and $\mathsf{c21}, N_2, C_2$. The bridge terms for the two direct paths shown in Fig. 2 are A and N_a. The adversary can always break concatenations down in this way:

Assumption 3 *If $p(i) \to p(i+1)$ is a bridge in bundle $\mathcal{B}$, then $\mathsf{msg}(p(i))$ is either an encryption or a basic value, but not a concatenation.*

For any bundle $\mathcal{C}$, there is an equivalent $\mathcal{B}$ in which the adversary separates every concatenated value to its basic or encrypted parts, and then subsequently reconcatenates these parts, as in Fig. 2, Part II [20, Prop. 9]. Assumption 3 restricts our attention to these equivalent but more convenient $\mathcal{B}$.

The direct paths form a framework that supports the extended paths:

Lemma 14. *Let $\mathcal{B}$ be a bundle, and p an extended penetrator path in $\mathcal{B}$ that is not direct. Let $p(i)$ be the earliest key node along p.*

1. *The part of p leading to $p(i)$ forms a direct path.*

2. *Let $p(j)$ be any key node along p, lying on an encryption or decryption strand s, $m_1 \Rightarrow p(j) \Rightarrow m_3$. There are direct paths q such that m_1, m_3 lie on q.*
3. *If s is an encryption strand, then $\mathsf{msg}(p(j)) \ll \mathsf{msg}(\mathsf{last}(q))$. If s is a decryption strand, then $\mathsf{msg}(p(j)) \ll \mathsf{msg}(\mathsf{first}(q))$.*

Proof. Claim 1 holds by the definition. Clause 2 is by induction on the $\preceq$-initial subgraphs of $\mathcal{B}$. Clause 3 follows by Lemma 13. □

We end our discussion of extended penetrator paths with a few key lemmas. The first says that a uniquely originating value that reaches a part of an execution must be transported there by a direct path. The second (an immediate consequence of Lemma 13) says that the adversary gains access to a basic value a along every direct path between endpoints that share no encryptions. This is important to us, because it means that the adversary can substitute a new value a' for a, causing a renaming at the end of the direct path, if nothing goes wrong later in the renamed bundle.

Lemma 15. *Let $\mathcal{B}$ be normal, and let S_1, S_2 partition $\mathcal{B}$ by strands. If a originates uniquely on $n_1 \in S_1$ and $a \ll \mathsf{msg}(n_2)$, where $n_2 \in S_2$, then there is a direct path p crossing from S_1 to S_2 such that $n_1 \preceq \mathsf{first}(p)$, $\mathsf{last}(p) \preceq n_2$, and $a \sqsubseteq \mathsf{bt}(p)$.*

Proof. First assume that $a \sqsubseteq \mathsf{msg}(n_2)$.

Let $M = \{m \in S_2 \colon a \sqsubseteq \mathsf{msg}(m) \text{ and } m \preceq n_2\}$. M is non-empty, so M has $\preceq$-minimal members; let m_0 be one of them. If m_0 is a transmission node, then a originates there, contrary to the assumption that it originates uniquely in S_1. So m_0 is a reception node. Let $P = \{p \colon \mathsf{last}(p) = m_0\}$. Suppose $a \sqsubseteq \mathsf{bt}(p)$ for some $p \in P$. By the minimality of m_0, $\mathsf{first}(p) \notin S_2$, so p is the desired path.

If, however, $a \not\sqsubseteq \mathsf{bt}(p)$ for all $p \in P$, then there must be an adversary node $\bullet \xrightarrow{a}$ originating a, contradicting unique origination on n_1.

Finally, if $a \ll \mathsf{msg}(n_2)$, but $a \not\sqsubseteq \mathsf{msg}(n_2)$, use Assumption 2 to obtain an earlier node $n_2' \Rightarrow n_2$ in which it is an ingredient, and apply the preceding. □

Corollary 1 *If, for all encryptions $\{|t|\}_K$, $\{|t|\}_K \sqsubseteq \mathsf{first}(p)$ implies $\{|t|\}_K \not\sqsubseteq \mathsf{last}(p)$, then $\mathsf{bt}(p)$ is a basic value.*

Lemma 16. *Let $\mathcal{B}$ be a normal bundle, and let p be an extended path in $\mathcal{B}$. If $p(j) \to p(j+1)$ is the last bridge along p, $\mathsf{msg}(p(j)) \ll \mathsf{msg}(\mathsf{last}(p))$.*

Proof. If $p(j), \ldots, \mathsf{last}(p)$ lies on a single direct penetrator path, then $\mathsf{msg}(p(j)) \sqsubseteq \mathsf{msg}(\mathsf{last}(p))$ by Lemma 13, whence $\mathsf{msg}(p(j)) \ll \mathsf{msg}(\mathsf{last}(p))$.

So suppose that the last bridge lies on an extended path that ends at a key node $p(k)$. Since there is no subsequent bridge, by the definition of normal bundle, this key node cannot lie on a destructive decryption strand; it must lie on a constructive encryption strand. Thus, $\mathsf{msg}(p(j)) \sqsubseteq \mathsf{msg}(p(k))$ and $\mathsf{msg}(p(k))$ is the key of $\mathsf{msg}(p(k+1)) = \{|t|\}_{\mathsf{msg}(p(k))}$. Since the remainder of p is constructive, $\mathsf{msg}(p(k+1)) \ll \mathsf{msg}(\mathsf{last}(p))$. □

The Separability Theorem. An extended path p is *critical* iff its source $\mathsf{first}(p)$ is not permitted to influence its target $\mathsf{last}(p)$.

We wish to remove the critical paths, since this will reduce a bundle to one that obeys the influence specification. If the adversary uses a path to influence a node, contrary to our $\rightsquigarrow$, we want to clip this path. If we can always remove these paths, and replace a Π-bundle containing critical paths with one with no critical paths, then even the adversary gets no advantage from critical paths. No violation of the influence specification is essential. Everything that can happen in Π can happen without violating the influence specification. If this is true in Π, we can assume $\rightsquigarrow$ when analyzing Π; nothing that matters will be left out.

A sufficient condition for this to hold is that Π's executions be "reparable:"

Definition 17. *A path p is* $\rightsquigarrow$-critical *in $\mathcal{B}$ iff* $\mathsf{first}(p) \not\rightsquigarrow \mathsf{last}(p)$.

$\mathcal{B}$ is $\rightsquigarrow$-reparable *iff $\rightsquigarrow$ is a may-influence relation for $\mathcal{B}$, and every $\rightsquigarrow$-critical path p has a bridge $p(i) \to p(i+1)$ where $\mathsf{msg}(p(i)) = a$ is a basic value.*

When $\rightsquigarrow$ is understood, we omit it and write "critical" or "reparable." We can assume no bridge term of p is a concatenation by Assumption 3. Thus, when p is reparable, $\mathsf{bt}(p)$ is a basic value. In Fig. 3, the most interesting bridge terms are N_1 and N_2, which are the uniquely originating values.

Theorem 18 (Separability). *For every $\rightsquigarrow$-reparable Π-bundle $\mathcal{B}$, there is a Π-bundle $\mathcal{C}$ lying below $\mathcal{B}$ such that $\mathcal{C}$ obeys $\rightsquigarrow$.*

Separability for protocols with session parameters. We will first apply Thm. 18 to the main case of protocols with session parameters, and $\rightsquigarrow_{\mathsf{s}}$. The key thing is to show that every critical path is of the first or second kind. The main reason why this is true is that—unless $\mathsf{first}(p) \rightsquigarrow_{\mathsf{s}} \mathsf{last}(p)$ and $\mathsf{last}(p) \rightsquigarrow_{\mathsf{s}} \mathsf{first}(p)$—all encryptions at the two ends contain different sets of session parameters. Thus, the bridge terms are basic values.

Theorem 19. *If Π is a protocol with session parameters, then every Π-bundle satisfying Φ_s is $\rightsquigarrow_{\mathsf{s}}$-reparable. Hence, by Thm. 18, Π obeys $\rightsquigarrow_{\mathsf{s}}$ subject to Φ_s.*

If each strand succeeds in choosing its session parameter freshly, then no two instances of the same role are related by the causal order in a reduced bundle, i.e. one obeying $\rightsquigarrow_{\mathsf{s}}$. This holds because any two instances supply different values for the session parameter, which are thus incompatible in $\rightsquigarrow_{\mathsf{s}}$.

Theorem 20. *Suppose that Π is a protocol with session parameters, and $\mathcal{B}$ obeys $\rightsquigarrow_{\mathsf{s}}$ and satisfies Φ_s. Then $s_1 \downarrow i \not\preceq s_2 \downarrow j$ when (i) $s_1 = \alpha(\rho)$ and $s_2 = \beta(\rho)$; and (ii) $P(\rho)$ is acquired on ρ by step $\min(i, j)$.*

5 Protocol Independence

We turn now from our focus on sessions to combining protocols. We organize the results by the choice of may-influence relation.

The discrete may-influence relation. Let Π_1 and Π_2 be protocols, i.e. sets of strands satisfying the assumptions mentioned in 2–3. For simplicity assume that the protocols are disjoint, in the sense that no strand (or initial segment) is an instance of a role of Π_1 and also an instance of a role of Π_2. Let $\Pi = \Pi_1 \cup \Pi_2$ be the protocol that contains all the roles of Π_1 and Π_2.

Define $n_1 \rightsquigarrow_1 n_2$ to hold for $n_1, n_2 \in \mathsf{regnodes}(\Pi)$ just in case $n_1, n_2 \in \mathsf{regnodes}(\Pi_1)$ or $n_1, n_2 \in \mathsf{regnodes}(\Pi_2)$. That is, nodes of the two source protocols may not influence each other.

We can use this *may-influence* relation to infer a protocol independence result, à la [1,9]. Define Π_1, Π_2 to have *sharply disjoint encryption* if

1. every key used for encryption on any node of either is a basic value; and
2. if e_1 is any encryption occurring in Π_1 and e_2 is any encryption occurring in Π_2, then e_1 and e_2 have no common instance.

The two conditions here are essentially syntactic. Condition 2 says that unification fails for the two encryptions. One way to satisfy condition 2 is using tags. If Π_1, Π_2 may have distinct tags τ_1, τ_2, such that every encryption in Π_i begins with tag τ_i, then condition 2 is certainly satisfied.

Theorem 21. *If Π_1, Π_2 have sharply disjoint encryption, then all $\Pi_1 \cup \Pi_2$ bundles are $\rightsquigarrow_1$-reparable. Hence, by Thm. 18, $\Pi_1 \cup \Pi_2$ obeys $\rightsquigarrow_1$.*

Proof. If p is a critical extended path, then the last bridge $p(k) \to p(k+1)$ on p either precedes a key node—which is a basic value by condition 1—or lies on an direct penetrator path from Π_i to Π_j. By Lemma 13, then, $\mathsf{bt}(p) \sqsubseteq \mathsf{first}(p)$ and $\mathsf{bt}(p) \sqsubseteq \mathsf{last}(p)$. Since these have no common encrypted ingredients, $\mathsf{bt}(p)$ is thus a basic value. $\square$

This is the essential idea behind [1,9]. The clever extension to algebras with convergent subterm rewrite rules in Ciobaca and Cortier's [8] appears to involve related ideas.

In our formalism, condition 1 is in fact unnecessary:

Theorem 22. *Let Π_1, Π_2 satisfy Condition 2 of sharply disjoint encryption, and let $\mathcal{B}$ be any bundle of $\Pi_1 \cup \Pi_2$. There is a bundle $\mathcal{C}$ lying below $\mathcal{B}$ such that $\mathcal{C}$ is $\rightsquigarrow_1$-reparable. Hence, by Thm. 18, $\Pi_1 \cup \Pi_2$ obeys $\rightsquigarrow_1$.*

The proof relies on two lemmas. If S is a set of nodes, let $\mathsf{enc}(S) = \{ \{\!|t|\!\}_K : \exists n \in S \;.\; \{\!|t|\!\}_K \ll \mathsf{msg}(n) \}$, the set of all encryptions occurring anywhere in the messages of S.

Lemma 23. *Let $\mathcal{B}$ be normal, $S, D \subseteq \mathsf{regnodes}(\mathcal{B})$ contain regular nodes, and let $n \in \mathsf{nodes}(\mathcal{B})$ be any node, such that $\mathsf{enc}(S) \cap \mathsf{enc}(D \cup \{n\}) = \emptyset$, $n \notin S$, and $\mathsf{bfringe}(n) \subseteq S \cup D$. Then, any crossing path p from S to $\{n\}$ has a bridge $p(k) \to p(k+1)$ where $\mathsf{msg}(p(k))$ is a basic value a, and either*

1. *$a \ll \mathsf{msg}(n)$; or else*

2. *After step k, there is a decryption strand s on which p joins a direct path q with $\mathsf{first}(q) \in D$. Moreover, $a \ll K^{-1}$ where K is the key in $\mathsf{msg}(s \downarrow 1) = \{\!|t|\!\}_K \in \mathsf{enc}(D)$.*

Proof. Let n be a reception node, since otherwise the claim is vacuous. We work by induction on the ordering $\preceq$. Our induction hypothesis is that the lemma holds for all nodes (whether regular or adversary nodes) $n_0 \prec n$ such that $\mathsf{bfringe}(n_0) \subseteq S \cup D$ for the same S, D.

If p is a direct path, then by Lemma 13, $\mathsf{bt}(p) \sqsubseteq \mathsf{first}(p)$ and $\mathsf{bt}(p) \sqsubseteq \mathsf{msg}(n)$, and since no encryption occurs in both, $\mathsf{bt}(p)$ is a basic value.

Otherwise, there is a key node along p; let $p(k+1)$ be the last one, i.e. the one with the greatest value of k. Then this key node is the middle node on an encryption or decryption strand $m \Rightarrow p(k+1) \Rightarrow p(k+2)$.

First, suppose that $m \Rightarrow p(k+1) \Rightarrow p(k+2)$ is an *encryption* strand. Since this is constructive and $\mathcal{B}$ is normal, $p(k+2), \ldots, \mathsf{last}(p)$ consists of *constructive* strands (encryptions and pairing) only. Thus $\mathsf{msg}(p(k+2)) \sqsubseteq \mathsf{msg}(n)$, and $\mathsf{msg}(p(k)) \ll \mathsf{msg}(n)$. Hence $\mathsf{msg}(p(k)) \in \mathsf{enc}(D \cup \{n\})$. Applying the induction hypothesis to $p(k)$, path p has the desired bridge no later than $p(k)$.

Next, suppose that $m \Rightarrow p(k+1) \Rightarrow p(k+2)$ is a *decryption* strand. By the definition of extended path, m must lie on a direct penetrator path q. The bridge of q must lie after the destructive strand $m \Rightarrow p(k) \Rightarrow p(k+1)$, and hence on the portion shared with p.

If $\mathsf{first}(q) \in S$, then by the case for a direct path, $\mathsf{bt}(q)$ satisfies the conditions.

Otherwise, $\mathsf{first}(q) \in D$. If $p(k)$ is a basic value, then condition 2 is satisfied. If $p(k)$ is an encryption t_2, then $\mathsf{msg}(m)$ is of the form $\{\!|t_1|\!\}_{t_2}$. By Lemma 13, $\{\!|t_1|\!\}_{t_2} \sqsubseteq \mathsf{first}(q)$, so that $t_2 \in \mathsf{enc}(D)$. Thus, the induction hypothesis, applied to $p(k)$, tells us that there is a basic value $a = p(i)$, where $i < k$, such that either $a \ll \mathsf{msg}(p(k))$, or else $a \ll K^{-1}$ for some $\{\!|t|\!\}_K \in \mathsf{enc}(D)$. In the first case, $a \ll \mathsf{msg}(p(k)) = t_2 \ll \{\!|t_1|\!\}_{t_2} \in \mathsf{enc}(D)$, as desired. In the second case, $p(i)$ and a satisfy the desired assertion. □

We may now return to the proof of Theorem 22.

Proof. Let p be a critical extended path, say with $\mathsf{first}(p) \in \mathsf{regnodes}(\Pi_1)$ and $\mathsf{last}(p) \in \mathsf{regnodes}(\Pi_2)$. Apply Lemma 23 with $S = \mathsf{regnodes}(\Pi_1) \cap \mathsf{bfringe}(\mathsf{first}(p))$ and $D = \mathsf{regnodes}(\Pi_2) \cap \mathsf{bfringe}(\mathsf{last}(p))$. Thus, p has a bridge $p(k) \to p(k+1)$ with a basic bridge term $\mathsf{msg}(p(k)) = a$.

We prepare $\mathcal{C}$ by breaking the paths for which condition 2 holds for a. If $a \notin \mathsf{unique}$ or if it originates on an adversary strand, we can immediately break the path.

Suppose $a \in \mathsf{unique}$ and originates on some $n_1 \in \mathsf{regnodes}(\Pi_1)$. Then by Assumption 2, clause 2, there is a reception node $m_0 \Rightarrow^+ \mathsf{last}(p)$ such that $a \sqsubseteq \mathsf{msg}(m_0)$. Hence, by Lemma 15, there is a direct path q crossing from $\mathsf{regnodes}(\Pi_1)$ to $\mathsf{regnodes}(\Pi_2)$ such that $\mathsf{last}(q) \preceq m_0$ and $a \sqsubseteq \mathsf{bt}(q)$. But since the endpoints have disjoint encryptions, $\mathsf{bt}(p) = a$. Thus, we can connect the bridge of q to $p(k+1)$.

Suppose $a \in \mathsf{unique}$ and originates on some $n_2 \in \mathsf{regnodes}(\Pi_2)$. Then (again via Lemma 15) there is a direct path q crossing from $\mathsf{regnodes}(\Pi_2)$ to $\mathsf{regnodes}(\Pi_1)$ such that $\mathsf{last}(q) \preceq \mathsf{first}(p)$ and $\mathsf{bt}(q) = a$. Thus, we can connect the bridge of q to $p(k+1)$. □

This shows a pitfall in interpreting strand-based results in the applied pi-calculus. In applied pi, letting $w = \mathsf{hash}(k_1, k_2)$, the two protocols P_1 and P_2:

$$P_1 = \nu\, k_1 s \;.\; \langle k_1 \rangle \;.\; \langle \{\!|\mathsf{t1}, s|\!\}_w \rangle \qquad P_2 = \nu\, k_2 s \;.\; \langle k_2 \rangle \;.\; \langle \{\!|\mathsf{t2}, s|\!\}_w \rangle$$

compose to yield $\nu\, k_1 k_2 s \;.\; P_1 \mid P_2$. In strands, by contrast, parameters in individual roles are essentially locally bound, since their possible instances are all substitution instances. Thus, there is no sense in which the two roles share the "same" k_1, k_2. Moreover, ν-binding expresses a notion of local choice that is somewhat different from both our unique origination and non-origination. It appears to be that the adversary never originates the ν-bound value. Thus, this result appears to be strong, but not truly comparable to results such as [9].

Another limitation of our result is that it is proved for a particular message algebra, and an adversary model for that, rather than for a class of algebras. We conjecture that there is a substantial class for which the lemmas of §§2, 4 hold, and that our results will hold throughout that class.

A one-way influence relation. Here we consider an asymmetric relation between the protocols Π_1, Π_2. Our goal is to ensure that adding the auxiliary protocol cannot undermine the main protocol Π_1. In many cases, Π_2 consumes cryptographically prepared units such as digital signatures or encrypted tickets (as in Kerberos), for instance, when it resumes sessions created by the main protocol. Thus, the main protocol may influence the auxiliary, but the reverse should not occur [19]. Let $\Pi = \Pi_1 \cup \Pi_2$, and define $n_1 \rightsquigarrow_2 n_2$ to hold for $n_1, n_2 \in \mathsf{regnodes}(\Pi)$ just in case $n_1 \in \mathsf{regnodes}(\Pi_1)$ or $n_2 \in \mathsf{regnodes}(\Pi_2)$.

With a more delicate definition of *disjoint encryption*, and stipulating the condition Φ that no uniquely originating value is contributed by Π_2, we obtain a properly stronger result. We say that t_0 is a *visible ingredient* in t_1, written $t_0 \sqsubseteq_v t_1$ iff t_1 may be obtained from t_0 by 0 or more pairings. That is, $\sqsubseteq_v$ is the smallest reflexive transitive relation such that $t \sqsubseteq_v t\,,\, t'$ and $t \sqsubseteq_v t'\,,\, t$.

Define Π_1 to have *disjoint encryption* from Π_2 iff:

1. If $\{\!|t|\!\}_K$ originates on any $n_2 \in \mathsf{regnodes}(\Pi_2)$, then $\{\!|t|\!\}_K \not\ll \mathsf{msg}(n_1)$, for any $n_1 \in \mathsf{regnodes}(\Pi_1)$.
   That is, encryptions created on Π_2 nodes are never accepted on Π_1 nodes.
2. Suppose $\{\!|t|\!\}_K$ originates on $n_1 \in \mathsf{regnodes}(\Pi_1)$, and $\{\!|t|\!\}_K \ll \mathsf{msg}(n_2)$ where $n_2 \in \mathsf{regnodes}(\Pi_2)$. If $t_0 \sqsubseteq t$ and $t_0 \sqsubseteq_v \mathsf{msg}(m_2)$, where $n_2 \Rightarrow^+ m_2$, then there is an m such that $m \Rightarrow^+ m_2$ and $t_0 \sqsubseteq_v \mathsf{msg}(m)$.
   That is, when Π_2 nodes receive Π_1 encryptions and extract new values from them, they never retransmit those values in visible form.

On these assumptions, the critical paths backward from $\Pi_2 \setminus \Pi_1$ to Π_1 may involve encryptions, but only if these were already transmitted forward, i.e. they are eliminated by Assumption 2. Similarly for fresh values originating on Π_1 nodes. This key idea in [19] is thus an instance of Thm. 18. There is however, one fine point. We must satisfy the "No open Vs" property of influence functions (Defn. 7, Cause 2), and this does not hold in all bundles. Namely, if a originates uniquely on a node $m_2 \in \mathsf{regnodes}(\Pi_2)$, and reaches a node $m_1 \in \mathsf{regnodes}(\Pi_1)$, it is permitted to flow back to a node $m_3 \in \mathsf{regnodes}(\Pi_2)$. To exclude these bundles, we use a constraint Φ. This constraint singles out the bundles in which no value required to be uniquely originating originates on a node of Π_2:

$$a \in \mathsf{unique} \text{ and } a \text{ originates on } n \text{ implies } n \notin \mathsf{regnodes}(\Pi_2). \qquad (\Phi)$$

Any bundle has a subbundle satisfying this constraint Φ, namely one that omits the offending values from unique. If any security goal of Π_1 has a counterexample, it has a counterexample satisfying Φ; cf. [17].

Theorem 24. *Let Π_1, Π_2 satisfy disjoint encryption, and let $\mathcal{B}$ be any bundle of $\Pi_1 \cup \Pi_2$ satisfying Φ. There is a bundle $\mathcal{C}$ lying below $\mathcal{B}$ such that $\mathcal{C}$ is $\rightsquigarrow_2$-reparable. Hence, by Thm. 18, $\Pi_1 \cup \Pi_2$ obeys $\rightsquigarrow_2$.*

Proof. If p is a critical path, then $\mathsf{first}(p) \in \mathsf{regnodes}(\Pi_2)$ and $\mathsf{last}(p) \in \mathsf{regnodes}(\Pi_1)$. Suppose $t \sqsubseteq_v \mathsf{first}(p)$ and $t \sqsubseteq t_1$, where t_1 is an encryption transmitted on a Π_1 node. Then by condition 2, t was received as a visible ingredient on some $m \Rightarrow \mathsf{first}(p)$. Thus, there is a bundle lying below $\mathcal{B}$ in which p receives this ingredient directly from some path leading to m.

Otherwise, we may apply Lemma 23 with $S = \mathsf{bfringe}(\mathsf{last}(p)) \cap \mathsf{regnodes}(\Pi_2)$ and $D = \mathsf{bfringe}(\mathsf{last}(p)) \cap \mathsf{regnodes}(\Pi_1)$. Condition 1 ensures the disjointness. We infer that there is a basic bridge term a along p. If $a \ll \mathsf{msg}(\mathsf{last}(p))$, then p is of the second kind.

Otherwise, a is a decryption key $a = K^{-1}$ for some $\{|t|\}_K \in \mathsf{enc}(D)$. If $a \notin \mathsf{unique}$, or if a originates on an adversary node, then we can immediately break p and connect its bridge to an adversary node.

Otherwise, if $a \in \mathsf{unique}$, then by constraint Φ, a originates uniquely on a regular node $n_1 \in \mathsf{regnodes}(\Pi_1)$. Using Lemma 15, there is a direct path q such that $\mathsf{last}(q) \in \mathsf{regnodes}(\Pi_1)$, $\mathsf{last}(q) \in \mathsf{regnodes}(\Pi_2)$, and $\mathsf{last}(q) \preceq \mathsf{first}(p)$. If $\mathsf{bt}(q)$ is an encryption, then we again use condition 2. If $\mathsf{bt}(q)$ is the basic value a, then p is again of the first kind. Hence we have a reparable bundle. $\square$

We may also use this second form of protocol independence to explain the "sequential composition" of Datta et al. [11]. Here, the nodes of the auxiliary protocol are placed after nodes of the primary protocol, but on the same strands; the formalization is unchanged. In particular, h maps nodes of the primary protocol to π_1 and nodes of the secondary protocol to π_2. The Clause 1 in Defn. 7 allows this to work when nodes of the secondary protocol never appear before a node of the primary protocol on any strand.

Vertical composition. Suppose that a protocol achieves a goal, assuming that it uses channels that provide particular kinds of protection against the adversary, e.g. that the adversary cannot spoof messages on these channels, or cannot snoop on their contents. Does that yield a secure protocol when these channels are replaced by subprotocols that ensure that the assumptions are met? This is the "vertical composition problem" [14,16,27]. Our methods seem highly relevant to this problem, but they require a way to express the channel assumptions as restrictions on the set of relevant bundles. We plan to explore this.

Conclusion. Two further main areas of future work remain. The more substantial is to adapt this approach to cover a notion of observational equivalence. This appears to involve enriching the adversary model to include a strand that detects the equality of two basic values. We also intend to soften the no Vs condition, which is tighter than necessary. For instance, it permits a tuple of messages to be received in a unit, but prohibits these same messages from being received successively, even when there are no intervening transmissions. More careful methods should relax this condition.

Acknowledgments. We are extremely grateful to Véronique Cortier, John Ramsdell, Paul Rowe, and the anonymous referees at POST.

References

1. S. Andova, C. Cremers, K. Gjøsteen, S. Mauw, S. Mjølsnes, and S. Radomirović. Sufficient conditions for composing security protocols. *Information and Computation*, 2007.
2. M. Arapinis, S. Delaune, and S. Kremer. From one session to many: Dynamic tags for security protocols. In *Proc. of LPAR'08*, 2008.
3. A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In *CAV*, pages 281–285, 2005.
4. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – Crypto '93 Proceedings*. Springer-Verlag, 1993.
5. K. Bhargavan, R. Corin, P.-M. Deniélou, C. Fournet, and J. J. Leifer. Cryptographic protocol synthesis and verification for multiparty sessions. In *IEEE Computer Security Foundations Symposium*, 2009.
6. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Technical Report 2000/067, IACR, Oct. 2001. Appeared in FOCS, 2001.
7. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology—EUROCRYPT 2001*, LNCS, pages 453–474. Springer, 2001.
8. Ş. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *CSF*, pages 322–336. IEEE Computer Society Press, July 2010.
9. V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, 2009.
10. V. Cortier, B. Warinschi, and E. Zalinescu. Synthesizing secure protocols. In *ESORICS: European Symposium On Research In Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 2007.

11. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13(3):423–482, 2005.
12. S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 239–251. IEEE Computer Society Press, June 2008.
13. P.-M. Deniélou and N. Yoshida. Dynamic multirole session types. In *POPL*, pages 435–446, 2011.
14. C. Dilloway and G. Lowe. Specifying secure transport channels. In *CSF*, pages 210–223. IEEE, 2008.
15. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
16. T. Groß and S. Modersheim. Vertical protocol composition. In *CSF*, pages 235–250. IEEE, 2011.
17. J. D. Guttman. Security goals and protocol transformations. In S. Mödersheim and C. Palamidessi, editors, *Tosca: Theory of Security and Applications*, LNCS. Springer, March 2011.
18. J. D. Guttman. Shapes: Surveying crypto protocol runs. In V. Cortier and S. Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, Cryptology and Information Security Series. IOS Press, 2011.
19. J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Computer Security Foundations Workshop*. IEEE CS Press, 2000.
20. J. D. Guttman and F. J. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002.
21. K. Honda, V. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138. Springer-Verlag, 1998.
22. K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *Proc. of POPL*, volume 43(1), pages 273–284. ACM, 2008.
23. A. Kamil and G. Lowe. Analysing TLS in the strand spaces model. *Journal of Computer Security*, 19(5):975–1025, 2011.
24. R. Küsters and M. Tuengerthal. Composition theorems without pre-established session identifiers. In *CCS*, pages 41–50. ACM, 2011.
25. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *LNCS*, pages 147–166, 1996.
26. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS*, pages 166–175. ACM, 2001.
27. S. Mödersheim and L. Viganò. Secure pseudonymous channels. *ESORICS*, pages 337–354, 2009.
28. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *CACM*, 21(12), Dec. 1978.
29. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Computer Security Foundations Workshop*, pages 174–, 2001.
30. F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.