Cryptographic Protocol Composition via the Authentication Tests[⋆]

Joshua D. Guttman

The MITRE Corporation
guttman@mitre.org
http://eprint.iacr.org/2008/430

Abstract. Although cryptographic protocols are typically analyzed in isolation, they are used in combinations. If a protocol Π_1, when analyzed alone, was shown to meet some security goals, will it still meet those goals when executed together with a second protocol Π_2? Not necessarily: for every Π_1, some Π_2s undermine its goals. We use the strand space "authentication test" principles to suggest a criterion to ensure a Π_2 preserves Π_1's goals; this criterion strengthens previous proposals.

Security goals for Π_1 are expressed in a language $\mathcal{L}(\Pi_1)$ in classical logic. Strand spaces provide the models for $\mathcal{L}(\Pi_1)$. Certain homomorphisms among models for $\mathcal{L}(\Pi)$ preserve the truth of the security goals. This gives a way to extract—from a counterexample to a goal that uses both protocols—a counterexample using only the first protocol. This model-theoretic technique, using homomorphisms among models to prove results about a syntactically defined set of formulas, appears to be novel for protocol analysis.

Protocol analysis usually focuses on the secrecy and authentication properties of individual, finished protocols. There is a good reason for this: Each security goal then definitely either holds or does not hold. However, the analysis is more reusable if we know which results will remain true after combination with other protocols, and perhaps other kinds of elaborations to the protocol.

In practice, every protocol is used in combination with other protocols, often with the same long-term keys. Also, many protocols contain messages with "blank slots." Higher level protocols piggyback on them, filling the blank spots with their own messages. We want to find out when the goals that hold of a protocol on its own are preserved under combination with other protocols, and when these blanks are filled in.

Two Results on Composition. Two existing results, both within the Dolev-Yao model [12], are particularly relevant. We showed [17] that if two protocols manipulate disjoint sets of ciphertexts, then combining the protocols cannot undermine their security goals. A careful, asymmetric formulation of this "disjoint encryption" property allowed us to show that one protocol Π_1 may produce

[⋆] Supported by MITRE-Sponsored Research.

L. de Alfaro (Ed.): FOSSACS 2009, LNCS 5504, pp. 303–317, 2009.

ciphertexts—in a broad sense including digital certificates as well as Kerberos-style tickets—consumed by another protocol Π_2, without Π_2 undermining any security goal of Π_1. The relation between Π_1 and Π_2 is *asymmetric* in that security goals of Π_2 could be affected by the behavior of Π_1, but not conversely.

Our result concerned only protocols that completely parse the messages they receive to atomic values, leaving no unstructured blank slots. A second limitation was to cover protocols using only atomic keys, not keys produced (e.g.) by hashing compound messages. A recent result by Delaune et al. [8] lifts these two limitations, but only in the *symmetric* case (akin to our Def. 9, clause 4). It applies only when neither protocol produces ciphertexts that may be consumed by the other, and hence when neither protocol could affect goals achieved by the other alone. Their method appears not to extend beyond this symmetric case.

One goal of this paper is an asymmetric result covering blank slots and compound keys.

Our Approach. Protocol executions—more specifically, the parts carried out by the rule-abiding, "regular" participants, but not the adversary—form objects we call *skeletons* [11]. A skeleton is *realized* if it contains enough protocol behavior so that, when combined with some adversary behavior, it can occur. If additional regular behavior is needed for a possible execution, then it is *unrealized.*

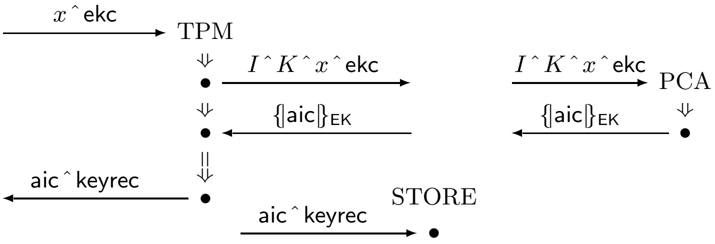
We introduce a new first order language $\mathcal{L}(\Pi)$ to describe skeletons of each protocol Π. Skeletons containing regular behaviors of Π provide a semantics, a set of models, for formulas of $\mathcal{L}(\Pi)$. Security goals are closed formulas $G \in \mathcal{L}(\Pi)$ of specific forms. A skeleton $\mathbb{A}$ is a *counterexample* to G when $\mathbb{A}$ is realized, but $\mathbb{A}$ satisfies G's negation, $\mathbb{A} \models \neg G$.

When Π_1 and Π_2 are protocols, $\mathcal{L}(\Pi_1)$ is a sublanguage of $\mathcal{L}(\Pi_1 \cup \Pi_2)$, and the security goals G_1 of $\mathcal{L}(\Pi_1)$ are some the goals of $\mathcal{L}(\Pi_1 \cup \Pi_2)$. The skeletons of Π_1 are those skeletons of $\Pi_1 \cup \Pi_2$ in which only Π_1 activity occurs.

We will define a syntactic relation between Π_1 and Π_2, called *strong disjoint encryption*, that ensures goals $G_1 \in \mathcal{L}(\Pi_1)$ are preserved. If any $\Pi_1 \cup \Pi_2$-skeleton is a counterexample to $G_1 \in \mathcal{L}(\Pi_1)$, we want to extract a Π_1-skeleton $\mathbb{A}_1$ which is a counterexample to G_1. Thus, Π_1 alone already undermines any goal that Π_1, Π_2 undermine jointly. The language $\mathcal{L}(\Pi)$, the definition of strong disjointness, and this result are the contributions of this paper.

The authentication test principles [10] suggest the definition of strong disjoint encryption, in two parts. First, Π_2 should not create encryptions of forms specified in Π_1; i.e. Π_2 should have no *creation conflicts.* Second, if a Π_2 execution receives a value only inside encryptions specified in Π_1, it should not re-transmit the value outside these encryptions; i.e. there should be no *extraction conflicts.*

We find Π_1-counterexamples from $\Pi_1 \cup \Pi_2$-counterexamples in two steps. First, we omit all non-Π_1 behavior. Second, we generalize: we remove all encrypted units not specified in Π_1 by inserting blank slots in their place. Each of these operations *preserves satisfaction* of $\neg G_1$. When Π_1, Π_2 has strong disjointness, they yield *realized* Π_1 skeletons from realized $\Pi_1 \cup \Pi_2$ skeletons. Hence, they preserve counterexamples.



$$\mathsf{ekc} = [\![\, ekc\ \mathsf{MF} \,\hat{}\, \mathsf{EK} \,]\!]_{\mathsf{sk(MF)}} \qquad \mathsf{aic} = [\![\, aic\ I \,\hat{}\, K \,\hat{}\, x \,]\!]_{\mathsf{sk(PCA)}}$$
$$\mathsf{keyrec} = \{\!|\, aikrec\ K, K^{-1} |\!\}_{\mathsf{SRK}}$$

Fig. 1. Modified Anonymous Identity Protocol

This reasoning is model-theoretic, characteristically combining two elements. One is the algebraic relations (embeddings and restrictions, homomorphisms, etc.) among the structures interpreting a logic. The second concerns syntax, often focusing on formulas of particular logical forms. A familiar example, combining these two ingredients, is the fact that a homomorphism between two structures for first order logic preserves satisfaction for atomic formulas.

A second goal of this paper is to illustrate this model-theoretic approach to security protocols.

Structure of This Paper. We first give an example certificate distribution protocol called MAIP. Section 1 gives background on strand spaces, including authentication tests. The goals of MAIP are formalized in $\mathcal{L}(\Pi)$, introduced (along with its semantics) in Section 2. Multiprotocols and strong disjointness are defined in Section 3. Section 4 gives the main results, and concludes.

Example: Anonymous Identities in Trusted Computing. A certificate distribution protocol (see Fig. 1) for "anonymous identity keys" is used with Trusted Platform Modules (TPMs). A Privacy Certificate Authority (PCA) creates a certificate aic binding a key K to a temporary name I. K is a public signature verification key. The certificate authorizes the signing key K^{-1} to sign requests on behalf of the holder of I [4].

Since the PCA knows nothing about the origin of a request, it transmits aic encrypted under key EK. The TPM manufacturer MF certifies in ekc that the matching private decryption key EK^{-1} resides within a TPM. If the request did not originate from this TPM, the certificate will never be decrypted, while otherwise that TPM will protect K^{-1} and use it according to certain rules. In particular, the TPM emits K, K^{-1} encrypted with a storage key SRK that only it possesses; this key record can be reloaded and used later.

We have omitted some details, and added a "blank slot" parameter x, which may be used to restrict the use of the aic. For instance, x can determine when the certificate expires, or it can limit its use to specific later protocols.

$[\![\, m \,]\!]_{\mathsf{sk}(A)}$ refers to m accompanied by a digitally signed hash, created with a signature key held by A. The tags *ekc*, *aic*, and *aikrec* are bitpatterns that distinguish units containing them from other cryptographically prepared values.

Security Goals of MAIP. MAIP has three main goals. They should hold whenever an $\mathsf{aic}, \mathsf{keyrec}$ pair is successfully stored for future use. First, the PCA should have produced aic, and transmitted it encrypted with *some* EK. Second, the TPM should have received aic encrypted with this EK, and retransmitted it in the clear. These goals are authentication goals, since they assert that uncompromised (regular) principals executed certain actions. The third is a confidentiality goal, stating that the private part K^{-1} of the AIK should never be observed, unprotected by encryption. Making our assumptions explicit, we get:

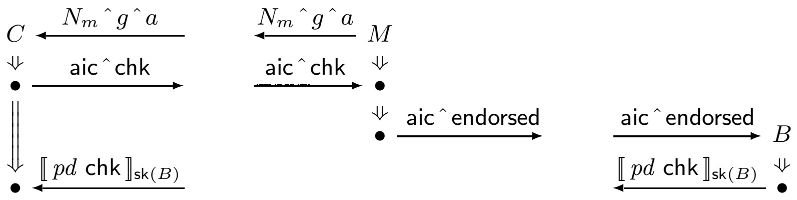
Whenever 1. STORE gets a message using parameters $I, K, x, \mathsf{PCA}, \mathsf{SRK}$;
2. $\mathsf{sk}(\mathsf{MF})$, $\mathsf{sk}(\mathsf{PCA})$, SRK^{-1} are used only in accordance with MAIP;
3. K, K^{-1} are generated only once,

then, for some public key EK,
1. A PCA run using the parameters $I, K, x, \mathsf{PCA}, \mathsf{EK}$ reached step 2;
2. A TPM run using the parameters $I, K, x, \mathsf{PCA}, \mathsf{EK}, \mathsf{SRK}$ reached step 3;
3. K^{-1} is never observed, unprotected by encryption.

Since the definitions of the protocol's roles are fixed, the goals do not need to say anything about the forms of the messages. They say only how far the role has progressed, and with what parameters. The unlinkability of K to a particular EK matches the classical existential quantifier used here.

MAIP and Other Protocols. A certificate distribution protocol like MAIP, on its own, is precisely useless.

MAIP becomes useful only if principals executing other protocols generate messages signed with K^{-1}, and accept messages verified with K, accompanied by a matching aic. For instance, a message signed with K^{-1} could be used to request access to network services, as is now widely done with Kerberos. More ambitiously, the aic could be regarded as a check-signing certificate from a bank's PCA. Then K^{-1} can be used to sign on-line checks that are anonymous to the payee, but guaranteed by the bank (Fig. 2). The blank slots g, a may be filled with formatted data representing the goods offered and amount to be paid.


C
$N_m \,\hat{}\, g \,\hat{}\, a$
$N_m \,\hat{}\, g \,\hat{}\, a$
M
aic ˆ chk
aic ˆ chk
aic ˆ endorsed
aic ˆ endorsed
B
$[\![\, pd \ \mathsf{chk} \,]\!]_{\mathsf{sk}(B)}$
$[\![\, pd \ \mathsf{chk} \,]\!]_{\mathsf{sk}(B)}$

$\mathsf{chk} = [\![\, chk\ I \,\hat{}\, M \,\hat{}\, a \,\hat{}\, N_m \,\hat{}\, \mathsf{hash}(g) \,]\!]_{K^{-1}}$ $\qquad$ $\mathsf{endorsed} = [\![\, ndrs\ \mathsf{chk} \,]\!]_{\mathsf{sk}(M)}$

Fig. 2. AIC-based check cashing

Unfortunately, the symmetric criterion for combining protocols [8] says nothing about how to construct such secondary protocols, since aic is a cryptographic unit that must be shared between the protocols. Our asymmetric criterion [17] does not apply, since MAIP, like many protocols, contains a blank slot x.

A criterion for safe composition should guide protocol designers to construct suites that work together. Our criterion says to avoid encryption creation conflicts and extraction conflicts. The protocol must not *create* anything unifying with the aic, ekc, and keyrec message formats, or with an aic encrypted with a public key. It must not *extract* anything that unifies with an aic from an encrypted item such as $\{\!|\mathsf{aic}|\!\}_{\mathsf{EK}}$. Creating and extracting units of these forms must remain the exclusive right of the primary protocol MAIP.

Our criterion of strongly disjoint encryption (Def. 9, Thm. 2) indicates that these are the only constraints on secondary protocols to interoperate with MAIP.

1 Messages, Protocols, Skeletons

Let $\mathfrak{A}_0$ be an algebra equipped with some operators and a set of homomorphisms $\eta\colon \mathfrak{A}_0 \to \mathfrak{A}_0$. We call members of $\mathfrak{A}_0$ *atoms.*

For the sake of definiteness, we will assume here that $\mathfrak{A}_0$ is the disjoint union of infinite sets of *nonces*, *atomic keys*, *names*, and *texts.* The operator $\mathsf{sk}(a)$ maps names to (atomic) signature keys, and K^{-1} maps an asymmetric atomic key to its inverse, and a symmetric atomic key to itself. Homomorphisms η are maps that respect sorts, and act homomorphically on $\mathsf{sk}(a)$ and K^{-1}.

Let X is an infinite set disjoint from $\mathfrak{A}_0$; its members—called *indeterminates*—act like unsorted variables. $\mathfrak{A}$ is freely generated from $\mathfrak{A}_0 \cup X$ by two operations: encryption $\{\!|t_0|\!\}_{t_1}$ and tagged concatenation $tag\ t_0\,\hat{}\,t_1$, where the tags tag are drawn from some set TAG. For a distinguished tag nil, we write $nil\ \ t_0\,\hat{}\,t_1$ as $t_0\,\hat{}\,t_1$ with no tag. In $\{\!|t_0|\!\}_{t_1}$, a non-atomic key t_1 is a symmetric key. Members of $\mathfrak{A}$ are called *messages.*

A homomorphism $\alpha = (\eta, \chi)\colon \mathfrak{A} \to \mathfrak{A}$ consists of a homomorphism η on atoms and a function $\chi\colon X \to \mathfrak{A}$. It is defined for all $t \in \mathfrak{A}$ by the conditions:

$$\begin{array}{llr} \alpha(a) = \eta(a), & \text{if } a \in \mathfrak{A}_0 & \alpha(\{\!|t_0|\!\}_{t_1}) = \{\!|\alpha(t_0)|\!\}_{\alpha(t_1)} \\ \alpha(x) = \chi(x), & \text{if } x \in X & \alpha(tag\ t_0\,\hat{}\,t_1) = tag\ \alpha(t_0)\,\hat{}\,\alpha(t_1) \end{array}$$

Indeterminates x serve as blank slots, to be filled by any $\chi(x) \in \mathfrak{A}$. This $\mathfrak{A}$ has the most general unifier property, which we will rely on. That is, suppose that for $v, w \in \mathfrak{A}$, there exist α, β such that $\alpha(v) = \beta(w)$. Then there are α_0, β_0, such that $\alpha_0(v) = \beta_0(w)$, and whenever $\alpha(v) = \beta(w)$, then α and β are of the forms $\gamma \circ \alpha_0$ and $\gamma \circ \beta_0$. Messages are abstract syntax trees in the usual way:

1. Let ℓ and r be the partial functions such that for $t = \{\!|t_1|\!\}_{t_2}$ or $t = tag\ t_1\,\hat{}\,t_2$, $\ell(t) = t_1$ and $r(t) = t_2$; and for $t \in \mathfrak{A}_0$, ℓ and r are undefined.
2. A *path* p is a sequence in $\{\ell, r\}^*$. We regard p as a partial function, where $\langle\rangle = \mathsf{Id}$ and $\mathsf{cons}(f, p) = p \circ f$. When the rhs is defined, we have: 1. $\langle\rangle(t) = t$; 2. $\mathsf{cons}(\ell, p)(t) = p(\ell(t))$; and 3. $\mathsf{cons}(r, p)(t) = p(r(t))$.
3. *p traverses a key edge* in t if $p_1(t)$ is an encryption, where $p = p_1{}^\frown\langle r\rangle{}^\frown p_2$.
4. *p traverses a member of S* if $p_1(t) \in S$, where $p = p_1{}^\frown p_2$ and $p_2 \neq \langle\rangle$.

5. t_0 *is an ingredient of* t, written $t_0 \sqsubseteq t$, if $t_0 = p(t)$ for some p that does not traverse a key edge in t.[1]
6. t_0 *appears in* t, written $t_0 \ll t$, if $t_0 = p(t)$ for some p.

A single local session of a protocol at a single principal is a *strand*, containing a linearly ordered sequence of transmissions and receptions that we call *nodes*. In Fig. 1, the vertical columns of nodes connected by double arrows $\Rightarrow$ are strands.

A message t_0 *originates* at a node n_1 if (1) n_1 is a transmission node; (2) $t_0 \sqsubseteq \mathsf{msg}(n_1)$; and (3) whenever $n_0 \Rightarrow^+ n_1$, $t_0 \not\sqsubseteq \mathsf{msg}(n_0)$.

Thus, t_0 originates when it was transmitted without having been either received or transmitted previously on the same strand. Values assumed to originate only on one node in an execution—*uniquely originating* values—formalize the idea of freshly chosen, unguessable values. Values assumed to originate nowhere may be used to encrypt or decrypt, but are never sent as message ingredients. They are called *non-originating* values. For a non-originating value K, $K \not\sqsubseteq t$ for any transmitted message t. However, $K \ll \{|t_0|\}_K \sqsubseteq t$ possibly, which is why we distinguish $\sqsubseteq$ from $\ll$. See [18,11] for more details.

In the tree model of messages, to apply a homomorphism, we walk through, copying the tree, but inserting $\alpha(a)$ every time an atom a is encountered, and inserting $\alpha(x)$ every time that an indeterminate x is encountered.

Definition 1. *Let S be a set of encryptions. A message t_0* is found only within *S* in *t_1, written $t_0 \odot^S t_1$, iff for every path p such that $p(t_1) = t_0$, either (1) p traverses a key edge or else (2) p traverses a member of S before its end.*

Message t_0 is found outside *S* in *t_1, written $t_0 \dagger^S t_1$, iff not $(t_0 \odot^S t_1)$.* □

Equivalently, $t_0 \dagger^S t_1$ iff for some path p, (1) $p(t_1) = t_0$, (2) p traverses no key edge, and (3) p traverses no member of S before its end. Thus, $t_0 \sqsubseteq t_1$ iff $t_0 \dagger^{\emptyset} t_1$.

E.g. $\mathsf{aic} \dagger^{\emptyset} \{|\mathsf{aic}|\}_{\mathsf{EK}}$, although $\mathsf{aic} \odot^{S_1} \{|\mathsf{aic}|\}_{\mathsf{EK}}$, where $S_1 = \{\, \{|\mathsf{aic}|\}_{\mathsf{EK}} \,\}$. The TPM transforms aic, transmitting a t such that $\mathsf{aic} \dagger^{S_1} t$, namely $t = \mathsf{aic} \hat{\ } \mathsf{keyrec}$.

Protocols. A *protocol Π* is a finite set of strands, representing the roles of the protocol. Three of the roles of the MAIP are the strands shown in Fig. 1. Their instances result by replacing I, K, etc., by any name, asymmetric key, etc., and replacing x by any (possibly compound) message. The fourth role is the *listener* role $\mathsf{Lsn}[y]$ with a single reception node in which y is received. The instances of $\mathsf{Lsn}[y]$ are used to document that values are available without cryptographic protection. For instance, $\mathsf{Lsn}[K]$ would document that K is compromised. Every protocol contains the role $\mathsf{Lsn}[y]$.

Indeterminates represent messages received from protocol peers, or passed down as parameters from higher-level protocols. Thus, we require:

If n_1 is a node on $\rho \in \Pi$, with an indeterminate $x \ll \mathsf{msg}(n_1)$,
then $\exists n_0$, $n_0 \Rightarrow^* n_1$, where n_0 is a reception node and $x \sqsubseteq \mathsf{msg}(n_0)$.

[1] $\sqsubseteq$ was formerly called the "subterm" relation [18], causing some confusion. A key is not an ingredient in its ciphertexts, but an aspect of how they were prepared, so $K \not\sqsubseteq \{|t|\}_K$ unless $K \sqsubseteq t$. Also, $\sqsubseteq \cap (\mathfrak{A}_0 \times \mathfrak{A}_0) = \mathsf{Id}_{\mathfrak{A}_0}$, so e.g. $a \not\sqsubseteq \mathsf{sk}(a)$.

So, an indeterminate is received as an ingredient before appearing in any other way. The initial node on the TPM AIC role in Fig. 1 shows x being received from the higher level protocol that has invoked the TPM activity.

A principal executing a role such as the PCA's role in MAIP may be partway through its run; for instance, it may have executed the first receive event without "yet" having executed its second event, the transmission node.

Definition 2. *Node n is a* role node *of Π if n lies on some $\rho \in \Pi$.*

Let n_j be a role node of Π of the form $n_1 \Rightarrow \ldots \Rightarrow n_j \Rightarrow \ldots$. Node m_j is an instance *of n_j if, for some homomorphism α, the strand of m_j, up to m_j, takes the form: $\alpha(n_1) \Rightarrow \ldots \Rightarrow \alpha(n_j) = m_j$.* □

That is, messages and their directions—transmission or reception—must agree up to node j. However, any remainders of the two strands beyond node j are unconstrained. They need not be compatible. When a protocol allows a principals to decide between different behaviors after step j, based on the message contents of their run, then this definition represents branching [14,16]. At step j, one doesn't yet know which branch will be taken.

Skeletons. A *skeleton* $\mathbb{A}$ consists of (possibly partially executed) role instances, i.e. a finite set of nodes, $\mathsf{nodes}(\mathbb{A})$, with two additional kinds of information:

1. A partial ordering $\preceq_{\mathbb{A}}$ on $\mathsf{nodes}(\mathbb{A})$;
2. Finite sets $\mathsf{unique}_{\mathbb{A}}, \mathsf{non}_{\mathbb{A}}$ of atomic values assumed uniquely originating and respectively non-originating in $\mathbb{A}$.

$\mathsf{nodes}(\mathbb{A})$ and $\preceq_{\mathbb{A}}$ must respect the strand order, i.e. if $n_1 \in \mathsf{nodes}(\mathbb{A})$ and $n_0 \Rightarrow n_1$, then $n_0 \in \mathsf{nodes}(\mathbb{A})$ and $n_0 \preceq_{\mathbb{A}} n_1$. If $a \in \mathsf{unique}_{\mathbb{A}}$, then a must originate at most once in $\mathsf{nodes}(\mathbb{A})$. If $a \in \mathsf{non}_{\mathbb{A}}$, then a must originate nowhere in $\mathsf{nodes}(\mathbb{A})$, though a or a^{-1} may be the key encrypting some $e \ll \mathsf{msg}(n)$ for $n \in \mathsf{nodes}(\mathbb{A})$.

$\mathbb{A}$ is *realized* if it is a possible run without additional activity of *regular* participants; i.e., for every reception node n, the adversary can construct $\mathsf{msg}(n)$ via the Dolev-Yao adversary actions,[2] using as inputs:

1. all messages $\mathsf{msg}(m)$ where $m \prec_{\mathbb{A}} n$ and m is a transmission node;
2. any atomic values a such that $a \notin (\mathsf{non}_{\mathbb{A}} \cup \mathsf{unique}_{\mathbb{A}})$, or such that $a \in \mathsf{unique}_{\mathbb{A}}$ but a originates nowhere in $\mathbb{A}$.

A homomorphism α yields a partial function on skeletons. We apply α to the messages on all nodes of $\mathbb{A}$, as well as to the sets $\mathsf{unique}_{\mathbb{A}}$ and $\mathsf{non}_{\mathbb{A}}$. We regard α as a homomorphism from $\mathbb{A}$ to $\alpha(\mathbb{A})$, when this is defined. However, α must not identify $K \in \mathsf{non}_{\mathbb{A}}$ with any atom that is an ingredient in any message in $\mathbb{A}$, or identify $a \in \mathsf{unique}_{\mathbb{A}}$ with another atom if this would give $\alpha(a)$ two originating nodes in $\alpha(\mathbb{A})$. A homomorphism α always acts as a bijection between $\mathsf{nodes}(\mathbb{A})$ and $\mathsf{nodes}(\alpha(\mathbb{A}))$. In [11] we use a compatible although more inclusive notion of homomorphism, since the action on nodes is not always bijective.

[2] The Dolev-Yao adversary actions are: concatenating messages and separating the pieces of a concatenation; encrypting a given plaintext using a given key; and decrypting a given ciphertext using the matching decryption key.

Authentication Tests. The core proof method in the strand space framework is the authentication test idea [11,10].[3] The idea concerns a realized skeleton $\mathbb{A}$ and a value c. If c was found only within a set of encryptions S, up to some point, and is later found outside S, then this "test" must be explained or "solved." Solutions are of two kinds: Either a key is compromised, so the adversary can create an occurrence of c outside S, or else a regular strand has a transmission node m_1 where, for all earlier nodes $m_0 \Rightarrow^+ m_1$,

$$c \odot^S \mathsf{msg}(m_0), \quad \text{but} \quad c \dagger^S \mathsf{msg}(m_1).$$

Since there are only finitely many roles in Π, unification on their nodes can find all candidates for regular solution nodes m_1. We formalize this using *cuts.*

Definition 3. *Let c be an atom or an encryption, and S be a set of encryptions.* $\mathsf{Cut}(c, S, \mathbb{A})$*, the* test cut *for c, S in $\mathbb{A}$, is defined if $\exists n_1 \in \mathsf{nodes}(\mathbb{A})$ such that $c \dagger^S \mathsf{msg}(n_1)$. In this case,*

$$\mathsf{Cut}(c, S, \mathbb{A}) = \{n \in \mathsf{nodes}(\mathbb{A}) \colon \exists m \,.\, m \preceq_{\mathbb{A}} n \wedge c \dagger^S \mathsf{msg}(m)\}. \qquad \square$$

For instance, in any skeleton $\mathbb{A}$ containing a full run of the TPM role, its fourth node n_4 is in the cut $\mathsf{Cut}(\mathsf{aic}, S, \mathbb{A})$ for *every* S, since n_4 transmits aic outside every encryption. Letting $\mathbb{A}_0$ be the skeleton containing all the activity in Fig. 1, with the visually apparent ordering, the third TPM node $n_3 \in \mathsf{Cut}(\mathsf{aic}, \emptyset, \mathbb{A})$ but $n_3 \notin \mathsf{Cut}(\mathsf{aic}, S_1, \mathbb{A})$ where $S_1 = \{ \{\!|\mathsf{aic}|\!\}_{\mathsf{EK}} \}$. In all $m \preceq_{\mathbb{A}_0} n_3$, $\mathsf{aic} \odot^{S_1} \mathsf{msg}(m)$.

Definition 4. *$U = \mathsf{Cut}(c, S, \mathbb{A})$ is* solved *if for every $\preceq_{\mathbb{A}}$-minimal $m_1 \in U$:*

1. *either m_1 is a transmission node;*
2. *or there is a listener node $m = \mathsf{Lsn}[t]$ with $m \prec_{\mathbb{A}} m_1$, and either*
   (a) $c = \{\!|t_0|\!\}_{t_1}$ and $t_1 = t$, or else
   (b) for some $\{\!|t_0|\!\}_{t_1} \in S$, t is the corresponding decryption key $t = t_1^{-1}$. $\square$

In the skeleton $\mathbb{A}_0$ from Fig. 1, the cut $\mathsf{Cut}(\mathsf{aic}, S_1, \mathbb{A})$ is solved by n_4. The cut $\mathsf{Cut}(\mathsf{aic}, \emptyset, \mathbb{A})$ is solved by the second PCA node, which transmits $\{\!|\mathsf{aic}|\!\}_{\mathsf{EK}}$, which means that it occurs outside the empty set at this point.

In MAIP, these are the two important cuts. In skeleton $\mathbb{A}_0$, they are solved by regular strands emitting the aic (clause 1) in new forms, but in other skeletons they could be solved by a listener node $m = \mathsf{Lsn}[\mathsf{privk}(PCA)]$ (clause 2a), or, for S_1, by $m' = \mathsf{Lsn}[\mathsf{EK}^{-1}]$ (clause 2b). In skeletons in which EK^{-1} and $\mathsf{privk}(PCA)$ are non-originating, then the TPM and PCA strands offer the only solutions.

Theorem 1 ([10]).

1. *If every well-defined cut in $\mathbb{A}$ is solved, $\mathbb{A}$ is realized.*
2. *If $\mathbb{A}$ is realized, then $\mathbb{A}$ has an extension $\mathbb{A}'$, obtained by adding only listener nodes, in which every well-defined cut is solved.*

Clauses 1 and 2 assert *completeness* [10, Prop. 5], and *soundness* [10, Props. 2,3], respectively.

[3] A fine point is that [11] worked in a framework without indeterminates X, while [10] established its completeness result for a stronger framework including them.

2 The Goal Language $\mathcal{L}(\Pi)$

$\mathcal{L}(\Pi)$ is a language for talking about executions of a protocol Π. We use typewriter font $\mathtt{x}, \mathtt{m}$, etc. for syntactic items including metasymbols such as $\mathtt{\Phi}, \mathtt{RhoJ}$.

Definition 5. *$\mathcal{L}(\Pi)$ is the classical quantified language with vocabulary:*

Variables *(unsorted) ranging over messages in $\mathfrak{A}$ and nodes;*
Function symbols $\mathtt{sk}$*,* $\mathtt{inv}$ *for the functions on $\mathfrak{A}_0$;*
Predicate symbols *equality* $\mathtt{u} = \mathtt{v}$*, falsehood* $\mathtt{false}$ *(no arguments), and:*

- $\mathtt{Non(v)}$*,* $\mathtt{Unq(v)}$*, and* $\mathtt{UnqAt(n, v)}$*;*
- $\mathtt{DblArrw(m, n)}$ *and* $\mathtt{Prec(m, n)}$*;*
- *One role predicate* $\mathtt{RhoJ}$ *for each role $\rho \in \Pi$ and each j with $1 \leq j \leq$* $\mathsf{length}(\rho)$*. The predicate* $\mathtt{RhoJ(m, v_1, \ldots, v_i)}$ *for the j^{th} node on ρ has as arguments: a variable* $\mathtt{m}$ *for the node, and variables for each of the i parameters that have appeared in any of ρ's first j messages.* □

Suppose for the moment that a message value v is associated with each variable $\mathtt{v}$, and the nodes m, n are associated with the variables $\mathtt{m}, \mathtt{n}$. Then the predicates $\mathtt{Non(v)}$, $\mathtt{Unq(v)}$, and $\mathtt{UnqAt(n, v)}$ are (respectively) true in a skeleton $\mathbb{A}$ when v is assumed non-originating in $\mathsf{non}_{\mathbb{A}}$; when v is assumed uniquely originating in $\mathsf{unique}_{\mathbb{A}}$; and when v is assumed uniquely originating in $\mathsf{unique}_{\mathbb{A}}$ and moreover originates at the node n in $\mathbb{A}$. The predicates $\mathtt{DblArrw(m, n)}$ and $\mathtt{Prec(m, n)}$ are (respectively) true in a skeleton $\mathbb{A}$ when the node m lies immediately before the node n, i.e. $m \Rightarrow n$; and when $m \prec_{\mathbb{A}} n$.

Role predicate $\mathtt{RhoJ(m, v_1, \ldots, v_i)}$ is true in a skeleton $\mathbb{A}$ when m is the j^{th} node of an instance of role ρ, with its parameters (in some conventional order) instantiated by the associated values $v_1, \ldots, v_i$. The role predicates are akin to the role state facts of multiset rewriting [13].

In particular, since every protocol Π contains the listener role $\mathsf{Lsn}[y]$, $\mathcal{L}(\Pi)$ always has a role predicate $\mathtt{Lsn1(m, x)}$, meaning that m is the first node on a listener strand receiving message x. It is used to express confidentiality goals.

The MAIP TPM role has four role predicates; the first two are:

- $\mathtt{maip\_tpm1(m, x)}$, meaning that m is a reception node not preceded by any other on its strand, and the message received is on node m is just the parameter x, as dictated by the definition of the MAIP TPM role;
- $\mathtt{maip\_tpm2(m, x, i, k, f, e)}$, meaning that m lies on the second position on its strand after a node m' such that $\mathtt{maip\_tpm1(m', x)}$, and m transmits message: $i \,\hat{}\, k \,\hat{}\, x \,\hat{}\, [\![\, ekc \;\; f \,\hat{}\, e \,]\!]_{\mathsf{sk}(f)}$. These are not independent; a valid formula is:

$$\mathtt{maip\_tpm2(m_2, x, i, k, f, e)} \supset \exists \mathtt{m_1} \,.\, \mathtt{DblArrw(m_1, m_2)} \wedge \mathtt{maip\_tpm1(m_1, x)}.$$

If Π_1 is a subprotocol of Π in the sense that every role of Π_1 is a role of Π, then $\mathcal{L}(\Pi_1)$ is a sublanguage of $\mathcal{L}(\Pi)$.

Two ingredients are conspicuously missing from $\mathcal{L}(\Pi)$. First, $\mathcal{L}(\Pi)$ has no function symbols for the constructors of $\mathfrak{A}$, namely encryption and concatenation. Second, $\mathcal{L}(\Pi)$ has no function which, given a node, would return the message sent or received on that node. We omitted them for two reasons.

$$\forall \mathtt{m}, \mathtt{I}, \mathtt{K}, \mathtt{x}, \mathtt{PCA}, \mathtt{MF}, \mathtt{SRK}\,.$$

$$\begin{aligned}
&\mathtt{if}\ \ \mathtt{Store1}(\mathtt{m}, \mathtt{I}, \mathtt{K}, \mathtt{x}, \mathtt{PCA}, \mathtt{SRK}) && (\Phi_1)\\
&\wedge\ \mathtt{Non}(\mathtt{skMF})\ \wedge\ \mathtt{Non}(\mathtt{skPCA})\ \wedge\ \mathtt{Non}(\mathtt{SRK}) && (\Phi_2)\\
&\wedge\ \mathtt{Unq}(\mathtt{K})\ \wedge\ \mathtt{Unq}(\mathtt{inv}(\mathtt{K})) && (\Phi_3)\\
&\mathtt{then}\ \exists \mathtt{n}_1, \mathtt{n}_2, \mathtt{EK}\,.\\
&\quad \mathtt{Pca2}(\mathtt{n}_1, \mathtt{I}, \mathtt{K}, \mathtt{x}, \mathtt{PCA}, \mathtt{EK})\ \wedge\ \mathtt{Tpm4}(\mathtt{n}_2, \mathtt{I}, \mathtt{K}, \mathtt{x}, \mathtt{PCA}, \mathtt{EK}, \mathtt{SRK}).
\end{aligned}$$

$$\forall \mathtt{m}, \mathtt{n}, \mathtt{I}, \mathtt{K}, \mathtt{x}, \mathtt{PCA}, \mathtt{MF}, \mathtt{SRK}\,.\ \mathtt{if}\ \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \mathtt{Lsn1}(\mathtt{n}, \mathtt{inv}(K))\ \mathtt{then}\ \mathtt{false}.$$

Fig. 3. Authentication and confidentiality goals in $\mathcal{L}(\Pi)$

First, the security goals we want to express need not be explicit about the forms of the messages sent and received. They need only refer to the underlying parameters. The definition of the protocol determines uniformly what the participants send and receive, as a function of these parameters. Moreover, assertions about compound messages embedded within parameters would provide artificial ways to construct counterexamples to our protocol independence theorem.

Second, $\mathcal{L}(\Pi)$ should be insensitive to the notational specifics of the protocol Π, describing the goals of the protocol without prejudicing the message syntax.

However, to reason axiomatically about protocols, we would work within an expanded language $\mathcal{L}'(\Pi)$ with message constructors for encryption and concatenation, and with a function to extract the message sent or received on a node. Goals would still be expressed in the sublanguage $\mathcal{L}(\Pi_1)$.

What Is a Security Goal? A security goal is either an authentication or a confidentiality property. An authentication goal requires a peer to have executed some behavior. A confidentiality goal requires some desired secret t not be shared as a parameter of another strand. Usually, this is a listener strand $\mathsf{Lsn}[t]$, so the goal ensures that t can never be transmitted unencrypted, in plaintext.[4]

Definition 6

1. *A* security claim *is a conjunction of atomic formulas of $\mathcal{L}(\Pi)$.*
2. *Suppose that G_0 is $\Phi \supset \exists \mathtt{v}_0 \ldots \mathtt{v}_\mathtt{j}\,.\,(\Psi_1 \vee \ldots \vee \Psi_\mathtt{k})$, where Φ and each $\Psi_\mathtt{i}$ is a security claim. Suppose that, for every variable $\mathtt{n}$ over nodes occurring free in G_0, some conjunct of Φ is a role predicate $\mathtt{RhoJ}(\mathtt{n}, \mathtt{u}, \ldots, \mathtt{w})$. Then the universal closure G of G_0 is a* security goal *of Π.*
3. *G is an authentication goal if $k > 0$ and a confidentiality goal if $k = 0$.* □

We identify the empty disjunction $\bigvee_{i\in\emptyset} \Psi_\mathtt{i}$ with $\mathtt{false}$. We identify the unit conjunction $\bigwedge_{i\in\{1\}} \Phi_\mathtt{i}$ with its sole conjunct $\Phi_\mathtt{i}$, and $\bigvee_{i\in\{1\}} \Phi_\mathtt{i}$ with $\Phi_\mathtt{i}$.

As examples, we formalize the authentication and confidentiality goals of Section 1 as two separate goals in Fig. 3. The authentication goal has a unit disjunction, i.e. Ψ_1 is everything inside the existential quantifier, and the confidentiality goal uses the vacuous disjunction $\mathtt{false}$, where $k = 0$.

[4] We consider only "full disclosure" goals, rather than "partial information" goals, in which a party learns that some values of t are possible, but not others. On the relation between full disclosure goals and partial information goals, see e.g. [3,7].

Semantics. The semantics for $\mathcal{L}(\Pi)$ are classical, with each structure a skeleton for the protocol Π. This requirement builds the permissible behaviors of Π directly into the semantics without requiring an explicit axiomatization.

Definition 7. *Let $\mathbb{A}$ be a skeleton for Π. An* assignment *σ for $\mathbb{A}$ is a function from variables of $\mathcal{L}(\Pi_1)$ to $\mathfrak{A} \cup \mathsf{nodes}(\Pi)$. Extend σ to terms of $\mathcal{L}(\Pi)$ via the rules: $\sigma(\mathtt{sk(t)}) = \mathsf{sk}(\sigma(\mathtt{t}))$, $\sigma(\mathtt{inv(t)}) = (\sigma(\mathtt{t}))^{-1}$.*

Satisfaction $\mathbb{A}, \sigma \models \Phi$ *is defined via the standard Tarski inductive clauses for the classical first order logical constants, and the base clauses:*

$\mathbb{A}, \sigma \models \mathtt{u} = \mathtt{v}$ *iff* $\sigma(\mathtt{u}) = \sigma(\mathtt{v})$*;*
$\mathbb{A}, \sigma \models \mathtt{Non(v)}$ *iff* $\sigma(\mathtt{v}) \in \mathsf{non}_{\mathbb{A}}$*;*
$\mathbb{A}, \sigma \models \mathtt{Unq(v)}$ *iff* $\sigma(\mathtt{v}) \in \mathsf{unique}_{\mathbb{A}}$*;*
$\mathbb{A}, \sigma \models \mathtt{UnqAt(m, v)}$ *iff* $\sigma(\mathtt{m}) \in \mathsf{nodes}(\mathbb{A})$*, and* $\sigma(\mathtt{v}) \in \mathsf{unique}_{\mathbb{A}}$*, and* $\sigma(\mathtt{v})$ *originates at node* $\sigma(\mathtt{m})$*;*
$\mathbb{A}, \sigma \models \mathtt{DblArrw(m, n)}$ *iff* $\sigma(\mathtt{m}), \sigma(\mathtt{n}) \in \mathsf{nodes}(\mathbb{A})$*, and* $\sigma(\mathtt{m}) \Rightarrow \sigma(\mathtt{n})$*;*
$\mathbb{A}, \sigma \models \mathtt{Prec(m, n)}$ *iff* $\sigma(\mathtt{m}) \prec_{\mathbb{A}} \sigma(\mathtt{n})$*;*

and, for each role $\rho \in \Pi$ and index j on ρ, the predicate $\mathtt{RhoJ}(\mathtt{m}, \mathtt{v}_1, \ldots, \mathtt{v}_\mathtt{k})$ *obeys the clause*

$\mathbb{A}, \sigma \models \mathtt{RhoJ}(\mathtt{m}, \mathtt{v}_1, \ldots, \mathtt{v}_\mathtt{k})$ *iff* $\sigma(\mathtt{m}) \in \mathsf{nodes}(\mathbb{A})$*, and*
$\sigma(\mathtt{m})$ *is an instance of the j^{th} node on role ρ,*
with the parameters $\sigma(\mathtt{v}_1), \ldots, \sigma(\mathtt{v}_\mathtt{k})$*.*

We write $\mathbb{A} \models \Phi$ when $\mathbb{A}, \sigma \models \Phi$ for all σ. □

When $\mathtt{n}$ is a variable over nodes, although $\sigma(\mathtt{n}) \notin \mathsf{nodes}(\mathbb{A})$ is permitted, in that case, whenever $\phi(\mathtt{n})$ is an atomic formula, $\mathbb{A}, \sigma \not\models \phi(\mathtt{n})$.

In protocols where there are two different roles ρ, ρ' that differ only after their first j nodes—typically, because they represent different choices at a branch point after the j^{th} node [16,14]—the two predicates $\mathtt{RhoJ}$ and $\mathtt{Rho'J}$ are equivalent.

Lemma 1. *If ϕ is an atomic formula and $\mathbb{A}$, $\sigma \models \phi$, then $\alpha(\mathbb{A})$, $\alpha \circ \sigma \models \phi$.*

If α is injective, and if ϕ is an atomic formula other than a role predicate $\mathtt{RhoJ}$*, and if $\alpha(\mathbb{A})$, $\alpha \circ \sigma \models \phi$, then $\mathbb{A}$, $\sigma \models \phi$.* □

3 Multiprotocols, Disjointness, and Authentication Tests

Given a primary protocol Π_1, as well as a protocol Π which includes it, we have written Π in the form $\Pi_1 \cup \Pi_2$, but this is imprecise. We distinguish the nodes of Π_1 from nodes of Π that do not belong to Π_1.

Definition 8

1. *(Π, Π_1) is a* multiprotocol *if Π, Π_1 are protocols, and every role of Π_1 is a role of Π.*
2. *Role node n_j is* primary *if it is an instance of a node of Π_1 (Def. 2). Role node n_2 is* secondary *if it is an instance of a node of Π, but it is not primary.*

3. *Instances of encryptions e_1 R-related to role nodes of Π_1 are in $E^R(\Pi_1)$:*

$$E^R(\Pi_1) = \{\alpha(e_1) \colon \exists n_1 \,.\, R(e_1, \mathsf{msg}(n_1)) \wedge n_1 \textit{ is a role node of } \Pi_1\}. \qquad \square$$

Below, we use the cases $\sqsubseteq$ and $\ll$ for R, i.e. $E^{\sqsubseteq}(\Pi_1)$ and $E^{\ll}(\Pi_1)$.

$E^{\ll}(\Pi_1) \neq \{e \colon \exists n_1, \alpha \,.\, e \ll \mathsf{msg}(\alpha(n_1)) \wedge n_1$ is a role node of $\Pi_1\}$, since the latter contains *all* encryptions, whenever any role of Π_1 uses an indeterminate (blank slots). $E^{\ll}(\Pi_1)$ requires that an encryption is *syntactically present* in a roles of Π_1, not instantiated from an indeterminate. The more naïve generalization of [17] would be useless for protocols with indeterminates. Refining the definition was easy, but proving it correct required a new method.

The secondary nodes of (Π, Π_1) do not form a protocol. Π_1 contains the listener role, so listener nodes are primary, not secondary. However, $(\Pi_1 \cup \Pi_2, \Pi_1)$ is a multiprotocol. Its secondary nodes are some of the instances of role nodes of Π_2, namely, those that are not also instances of role nodes of Π_1.

Strong Disjointness. To ensure that a Π does not interfere with the goals of Π_1, we control how the secondary nodes transform encryptions. To create an encryption is one way to transform it, or another way is to extract some ingredient—such as a smaller encryption or a nonce or key—from inside it.

Definition 9

1. *If any $e \in E^{\ll}(\Pi_1)$ originates on a secondary transmission node n_2, then n_2 is an* encryption creation conflict.
2. *A secondary transmission node n is an* extraction conflict *if $t_1 \dagger^S \mathsf{msg}(n)$ for some $S \subseteq E^{\sqsubseteq}(\Pi_1)$ where $t_1 \sqsubseteq e \in S$, and:*

$$(\exists m \,.\, m \Rightarrow^+ n \;\wedge\; e \sqsubseteq \mathsf{msg}(m)) \quad \wedge \quad (\forall m \,.\, m \Rightarrow^+ n \;\supset\; t_1 \odot^S \mathsf{msg}(m)).$$

3. *Π, Π_1 has* strongly disjoint encryption *(s.d.e.) iff it has neither encryption creation conflicts nor extraction conflicts.*
4. *Π_1 and Π_2 are* symmetrically disjoint *if, letting $\Pi = \Pi_1 \cup \Pi_2$, both Π, Π_1 and Π, Π_2 have s.d.e.* □

Creation conflicts and extraction conflicts are the two ways that Π could create new ways to solve authentication tests already present in Π_1. Thus, s.d.e. ensures that only Π_1 solutions are needed for the tests in a Π_1 skeleton.

Strong disjointness is a syntactic property, even though its definition talks about all strands of the protocols Π and Π_1. We can check it using unification, as, in Figs. 1–2, we can quickly observe that the check-cashing protocol never creates an ekc, aic, or keyrec, and never extracts an aic from an encryption. Protocol analysis tools such as CPSA [11] can be programmed to check for it.

4 Protocol Independence

For any goal $G_1 \in \mathcal{L}(\Pi_1)$, we want to squeeze a Π_1-counterexample $\mathbb{A}_1$ out of a Π-counterexample $\mathbb{B}$. We do this in two steps: First, we restrict $\mathbb{B}$ to its primary nodes $\mathbb{B}\restriction\Pi_1$. Then, we remove all non-primary encryptions $e_2 \notin E^{\ll}(\Pi_1)$ from $\mathbb{B}\restriction\Pi_1$, by replacing them with indeterminates. In the reverse direction, this is a homomorphism. I.e., there is a $\mathbb{A}_1$ and a homomorphism α such that no secondary encryptions e_2 appear in $\mathbb{A}_1$, and $\mathbb{B}\restriction\Pi_1 = \alpha(\mathbb{A}_1)$. We call this second step "removal."

Definition 10. *Let $\{e_i\}_{i\in I}$ be the indexed family of all secondary encryptions appearing in a Π-skeleton $\mathbb{B}$, without repetitions, and let $\{x_i\}_{i\in I}$ be an indexed family of indeterminates without repetitions, none of which appear in $\mathbb{B}$.*

The homomorphism α that maps $x_i \mapsto e_i$, and is the identity for all atoms and all indeterminates not in $\{x_i\}_{i\in I}$ is a removal *for $\mathbb{B}$.* □

For a removal α, there are $\mathbb{A}$s with $\mathbb{B} = \alpha(\mathbb{A})$. To compute a canonical one, for each $n \in \mathsf{nodes}(\mathbb{B})$, we walk the tree of $\mathsf{msg}(n)$ from the top. We copy structure until we reach a subtree equal to any e_i, when we insert x_i instead. The resulting $\mathbb{A}$ is the *result* of the removal α for $\mathbb{B}$. The *result* of α for a node $m \in \mathbb{B}$ means the $n \in \mathsf{nodes}(\mathbb{A})$ such that $\alpha(n) = m$.

Lemma 2. *Let Π, Π_1 be a multiprotocol, with $G_1 \in \mathcal{L}(\Pi_1)$ a Π_1 security goal.*

1. *If $\mathbb{B} \models \neg G_1$, then $\mathbb{B}\restriction\Pi_1 \models \neg G_1$.*
2. *Let α be a removal for $\mathbb{B}$ with result $\mathbb{A}$.*
   (a) *If $\alpha(n)$ is a primary node, then n is a primary node for the same role.*
   (b) *If ϕ is a role predicate and $\mathbb{B}, \alpha \circ \sigma \models \phi$, then $\mathbb{A}, \sigma \models \phi$.*
   (c) *If $\mathbb{B} \models \neg G_1$, then $\mathbb{A} \models \neg G_1$.*

The definition of strong disjointness (Def. 9) implies, using Thm. 1:

Lemma 3. *Let Π, Π_1 have s.d.e., and let α be a removal for $\mathbb{B}\restriction\Pi_1$ with result $\mathbb{A}$. If $\mathbb{B}$ is a realized Π-skeleton, then $\mathbb{A}$ is a realized Π_1-skeleton.*

The essential idea here is to show that a solution to a primary cut lies on a primary node, which will be preserved in the restriction, and then again preserved by the removal. From the two preceding results, we obtain our main theorem:

Theorem 2. *Let Π, Π_1 have s.d.e., and let $G_1 \in \mathcal{L}(\Pi_1)$ be a security goal. If $\mathbb{A} \models \neg G_1$ and is realized, then for some realized Π_1-skeleton $\mathbb{A}_1$, $\mathbb{A}_1 \models \neg G_1$.*

Thm. 2 implies, for instance, that the check-cashing protocol of Fig. 2 preserves the goals of MAIP.

Conclusion. Our result Thm. 2 uses a new, model-theoretic approach. It combines reasoning about the logical form of formulas—the security goals G—with operations on the structures that furnish models of these formulas. These operations are restriction, homomorphisms, and removals. The authentication tests suggest the definition of strong disjointness (Def. 9).

Thm. 2 simplifies establishing that two protocols combine to achieve their goals. Goals of the joint protocol Π expressed in $\mathcal{L}(\Pi_1)$ may be verified without reference to $\Pi \setminus \Pi_1$. Second, our composition result can also be read as a prescription—or a heuristic—for protocol design. Protocols can be built from subprotocols that provide some of the intermediate cryptographic values that they require. Thm. 2 gives the constraints that a protocol designer must adhere to, in enriching an existing suite of protocols. His new operations must be strongly disjoint from the existing protocols, regarded as a primary protocol.

Related Work. An outstanding group of articles by Datta, Derek, Mitchell, and Pavlovic, including [9], concern protocol derivation and composition. The authors explore a variety of protocols with common ingredients, showing how they form a sort of family tree, related by a number of operations on protocols.

Our definition of multiprotocol covers both [9]'s *parallel composition* and its *sequential composition. Refinement* enriches the message structure of a protocol. *Transformation* moves information between protocol messages, either to reduce the number of messages or to provide a tighter binding among parameters.

Despite their rich palette of operations, their main results are restricted to parallel and sequential composition [9, Thms. 4.4, 4.8]. Each result applies to particular proofs of particular security goals G_1. Each proof relies on a set Γ of invariant formulas that Π_1 preserves. If a secondary protocol Π_2 respects Γ, then G_1 holds of the parallel composition $\Pi_1 \cup \Pi_2$ (Thm. 4.4). Thm 4.8, on sequential composition, is more elaborate but comparable. By contrast, our Thm. 2 is one uniform assertion about all security goals, independent of their proofs. It ensures that Π_2 will respect all usable invariants of Π_1. This syntactic property, checked once, suffices permanently, without looking for invariants to re-establish.

Universal composability [6] is a related property, although expressed in a very different, very strong, underlying model. It is often implemented by randomly choosing a tag to insert in all messages of a protocol, this tag being chosen at session set-up time. Thus, the symmetric disjointness of any two sessions, whether of the same or of different protocols, holds with overwhelming probability.

Andova, et al. [1] study sequential and parallel composition, using tags or distinct keys as implementation strategies, as in our [17]. They independently propose a definition [1, Def. 25] like the symmetric definition of [8].

Related but narrower problems arise from *type-flaw attacks*, situations in which a participant may parse a message incorrectly, and therefore process it in inappropriate ways [19]. Type flaw attacks concern a single protocol, although a protocol that may be viewed with different degrees of explicit tagging.

Future Work. Is there a theorem like Thm. 2 for the refinement and transformation operations [9]? For specific, limited formulations, the answer should be affirmative, and the model-theoretic approach is promising for establishing that answer. Such a result would provide a strong guide for protocol design.

Acknowledgments. I am grateful to Stéphanie Delaune and to my colleagues Leonard G. Monk, John D. Ramsdell, and F. Javier Thayer. An extremely perceptive anonymous referee report from CSF provoked a radical reworking. MITRE release number: 07-0029.

References

1. Andova, S., Cremers, C.J.F., Gjøsteen, K., Mauw, S., Mjølsnes, S.F., Radomirović, S.: Sufficient conditions for composing security protocols. Information and Computation (2007)
2. Backes, M., Maffei, M., Unruh, D.: Zero-knowledge in the applied pi-calculus and automated verification of the Direct Anonymous Attestation protocol. In: IEEE Symposium on Security and Privacy (2008)
3. Backes, M., Pfitzmann, B.: Relating cryptographic and symbolic key secrecy. In: Proceedings of 26th IEEE Symposium on Security and Privacy (May 2005)
4. Balacheff, B., Chen, L., Pearson, S., Plaquin, D., Proudler, G.: Trusted Computing Platforms: TCPA Technology in Context. Prentice Hall PTR, NJ (2003)
5. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM Conference on Communications and Computer Security (CCS) (2004)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, IACR 2000/067 (October 2001)
7. Canetti, R., Herzog, J.: Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 380–403. Springer, Heidelberg (2006)
8. Cortier, V., Delaitre, J., Delaune, S.: Safely composing security protocols. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 352–363. Springer, Heidelberg (2007)
9. Datta, A., Derek, A., Mitchell, J.C., Pavlovic, D.: A derivation system and compositional logic for security protocols. Journal of Computer Security 13(3), 423–482 (2005)
10. Doghmi, S.F., Guttman, J.D., Thayer, F.J.: Completeness of the authentication tests. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 106–121. Springer, Heidelberg (2007)
11. Doghmi, S.F., Guttman, J.D., Thayer, F.J.: Searching for shapes in cryptographic protocols. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 523–537. Springer, Heidelberg (2007), Extended version, http://eprint.iacr.org/2006/435
12. Dolev, D., Yao, A.: On the security of public-key protocols. IEEE Transactions on Information Theory 29, 198–208 (1983)
13. Durgin, N., Lincoln, P., Mitchell, J., Scedrov, A.: Multiset rewriting and the complexity of bounded security protocols. Journal of Computer Security 12(2), 247–311 (2004)
14. Fröschle, S.: Adding branching to the strand space model. In: Proceedings of EXPRESS 2008. Electronic Notes in Theoretical Computer Science. Elsevier, Amsterdam (2008)
15. Goguen, J.A., Meseguer, J.: Order-sorted algebra I. Theoretical Computer Science 105(2), 217–273 (1992)
16. Guttman, J.D., Herzog, J.C., Ramsdell, J.D., Sniffen, B.T.: Programming cryptographic protocols. In: De Nicola, R., Sangiorgi, D. (eds.) TGC 2005. LNCS, vol. 3705, pp. 116–145. Springer, Heidelberg (2005)
17. Guttman, J.D., Thayer, F.J.: Protocol independence through disjoint encryption. In: Proceedings of 13th Computer Security Foundations Workshop. IEEE Computer Society Press, Los Alamitos (2000)
18. Guttman, J.D., Thayer, F.J.: Authentication tests and the structure of bundles. Theoretical Computer Science 283(2), 333–380 (2002)
19. Heather, J., Lowe, G., Schneider, S.: How to prevent type flaw attacks on security protocols. In: Proceedings of 13th Computer Security Foundations Workshop. IEEE Computer Society Press, Los Alamitos (2000)