A Generalized Two-Phase Analysis of Knowledge Flows in Security Protocols

Marten van Dijk Emina Torlak Blaise Gassend
Srinivas Devadas
MIT Computer Science and Artificial Intelligence laboratory, Cambridge, USA
{marten, emina, gassend, devadas}@mit.edu

Abstract

We introduce *knowledge flow analysis*, a simple and flexible formalism for checking cryptographic protocols. Knowledge flows provide a uniform language for expressing the actions of principals, assumptions about intruders, and the properties of cryptographic primitives. Our approach enables a generalized two-phase analysis: we extend the two-phase theory by identifying the necessary and sufficient properties of a broad class of cryptographic primitives for which the theory holds. We also contribute a library of standard primitives and show that they satisfy our criteria.

keywords: security protocols, intruder detection.

1 Introduction

One area of major successes for formal methods has been the verification of security protocols. A number of specialized tools have been developed in the last decade that have exposed subtle flaws in existing protocols (see, e.g. [27, 11]). Many of these tools [10, 11, 15] use a two-phase approach to efficiently identify intrusion scenarios.

This paper presents *knowledge flow analysis*, a lightweight and flexible formalism for checking cryptographic protocols. Our approach is based on a simple mathematical foundation that provides an extensible framework for two-phase analysis. In particular, we generalize the two-phase theory of Clarke et al [10] by identifying the properties of cryptographic primitives for which the theory holds. We demonstrate the generality of our criteria by using them to build a library of standard cryptographic primitives: public and symmetric encryption/decryption, signing, pair/set construction, nonce generation, and hashing. The sample library can be extended to include blind signing, certification, and many other functions from the rich class of primitives that satisfy our criteria.

Our approach gives a uniform formalism for expressing the actions of principals, assumptions on intruders, and properties of cryptographic primitives. The dynamic behavior of the protocol is described by an initial state of knowledge, and a collection of rules that dictate how knowledge may flow amongst principals. Protocol rules are embedded into the initial state of knowledge as values that can be composed and decomposed by a special *rule primitive*, which satisfies the two-phase criteria.

The knowledge flow approach grew out of an effort to check a new cryptographic scheme [19, 18]. Knowledge flow analysis described here was the final result of a series of incremental attempts at formalizing and checking their assumptions using the Alloy language and tool [24, 23]. This process drew out *the single source axiom* which, to the best of our knowledge, has not been described before: the security of cryptographic functions depends on the assumption that their fixed points are hard to compute.

The rest of the paper is structured as follows. Section 2 explains the key intuitions underlying the approach, using the Needham-Schroeder protocol [34] as an example. Section 3 gives the mathematical foundations of the approach. Section 4 formulates and proves the two-phase theory in the knowledge flow context. Section 5 provides a mathematical characterization of the primitives to which the two-phase theory applies. Section 6 presents a sample subset of these primitives. The paper closes with a discussion of related work and concluding remarks.

2 Knowledge Flow Basics

The key idea behind knowledge flow analysis is the observation that, at the most basic level, the purpose of a security protocol is to distribute knowledge among its legitimate participants. A protocol is flawed if it allows an intruder to learn a value that is intended to remain strictly within the legitimate principals' pool of knowledge. To gain more intuition about knowledge flows in security applications, consider the original Needham-Schroeder protocol [34]:

1. Alice transmits $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$ to Bob
2. Bob transmits $\mathcal{E}_{\mathcal{PK}(A)}(N_A, N_B)$ to Alice
3. Alice transmits $\mathcal{E}_{\mathcal{PK}(B)}(N_B)$ to Bob.

We have two principals, Alice and Bob, each of whom has an initial supply of knowledge. Alice's initial knowledge, for example, consists of her own public/private key pair $\mathcal{PK}(A)/\mathcal{SK}(A)$, identity I_A, nonce N_A, and Bob's public key $\mathcal{PK}(B)$ and identity I_B. The purpose of the protocol is to distribute the nonces between Alice and Bob in such a way that the following conditions hold at the end: (i) Alice and Bob both know N_A and N_B, and (ii) no other principal knows both nonces.

To initiate the protocol, Alice first expands her pool of knowledge to include $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$, an encryption of her identity and nonce with Bob's public key. She then sends the cipher to Bob who decrypts it using his private key, $\mathcal{SK}(B)$. At the end of the first step of the protocol, Bob's knowledge has increased to include the values $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$ and N_A. Bob performs the second step of the protocol by adding $\mathcal{E}_{\mathcal{PK}(A)}(N_A, N_B)$ to his current knowledge and sending the cipher to Alice. She uses her private key to decrypt Bob's message and extract N_B. By using N_B and $PK(B)$, Alice can set up an authenticated and private channel with Bob as is done during the final step of the protocol in which Alice creates $\mathcal{E}_{\mathcal{PK}(B)}(N_B)$ and forwards it to Bob. Both Alice and Bob now know the two nonces and share all the knowledge except their secret keys.

Following the flow of knowledge in the Needham-Schroeder protocol provides a crucial insight underlying our analysis method. Namely, a principal can learn a value in one of three ways; he can

- *draw* the value at the start,
- *compute* it using his current knowledge, or
- *learn* it by communication.

Our analysis treats the latter two ways of obtaining knowledge as equivalent. Specifically, we can think of Alice's computing $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$ as her learning it from a principal called *Encryptor* whose initial pool of values includes all possible ciphers: Alice sends the tuple $(\mathcal{PK}(B), (I_A, N_A))$ to *Encryptor* who responds by sending back the encryption of (I_A, N_A) with $\mathcal{PK}(B)$.

Treating cryptographic primitives as principals allows us to consider the total pool of knowledge to be *fixed*. That is, the set of all values before and after the execution of a security protocol is the same; the only difference is the distribution of those values among the principals. Since we assume that principals never

forget values, the set of principals who know a value at the end of a protocol session subsumes the set of principals who drew the value at the beginning.

The goal of analyzing knowledge flows in a protocol is to verify that particular values never leak out of the honest participants' pool of knowledge. In other words, *we are interested in analyzing the flow of knowledge from an intruder's perspective.* This observation allows us to make sound simplifying assumptions that drastically reduce the effort needed to formalize a protocol in terms of knowledge flows:

- We need not encode the flows of knowledge among the honest principals, such as the flow which allows Alice to learn $\mathcal{E}_{\mathcal{PK}(A)}(N_A, N_B)$ from $Encryptor$. Rather, we may assume that each honest principal draws all values in the total knowledge pool and specify protocols solely in terms of the intruders' knowledge flows (sections 3.1 and 3.2).
- We may model all adversaries, including the untrusted public network, with a single opponent whom we call $Oscar$. The soundness of this approach is formally proved in section 3.3. Intuitively, the approach makes sense if we note that the potential adversaries will be most effective when they collaborate and share knowledge among themselves. Hence, we can replace the (collaboration of) adversaries with a single principal who possesses all their knowledge, without excluding any intrusion scenarios.

In our example, the flow of knowledge from the intruder's perspective starts with the protocol initialization message $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$, since Oscar needs no prior knowledge to learn the first cipher that Alice sends to Bob. In general, because Oscar includes the untrusted public network, he learns the first message of the protocol for free, regardless of who its intended recipient and sender are:

$$\forall_{p\in\{a,b\},p'\in\{a,b\}\cup O}\ [\emptyset \rightarrow \mathcal{E}_{\mathcal{PK}(p')}(\mathcal{I}(p), \mathcal{N}(\epsilon, \mathcal{I}(p)))]. \tag{1}$$

The variables a and b denote the honest principals (Alice and Bob), and the set O stands for Oscar. The notation $\mathcal{N}(\epsilon, \mathcal{I}(p))$ represents the nonce that the nonce primitive generated for the principal identified by $\mathcal{I}(p)$ using the random value ϵ as the seed. For example, Alice's identity is $\mathcal{I}(a) = I_A$ and Alice's nonce is $\mathcal{N}(\epsilon, \mathcal{I}(a)) = N_A$. The empty set means that Oscar needs no prior knowledge to learn $\mathcal{E}_{\mathcal{PK}(p')}(\mathcal{I}(p), \mathcal{N}(\epsilon, \mathcal{I}(p)))$.

Once his pool of knowledge includes $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$, Oscar learns the corresponding response, $\mathcal{E}_{\mathcal{PK}(A)}(N_A, N_B)$. More generally[1],

$$\forall_{p'\in\{a,b\},p\in\{a,b\}\cup O,v\in V}\ \left[\{c\} \rightarrow \mathcal{E}_{\mathcal{PK}(p)}(v, \mathcal{N}(c, \mathcal{I}(p')))\right] \text{ where } c = \mathcal{E}_{\mathcal{PK}(p')}(\mathcal{I}(p), v). \tag{2}$$

The variable V denotes the set of all values, or the fixed pool of knowledge. Note that our formalization constrains the seed of Bob's nonce to be Alice's initialization message. This is needed to establish that Bob's nonce was generated in the context of the protocol session started by Alice with $\mathcal{E}_{\mathcal{PK}(B)}(I_A, N_A)$. The resulting correspondence between the nonces prevents our analysis from sounding false alarms when Oscar legitimately obtains two nonces from Alice and Bob by running a valid protocol session with each.

Oscar learns the final message, $\mathcal{E}_{\mathcal{PK}(B)}(N_B)$, as a consequence of knowing $\mathcal{E}_{\mathcal{PK}(A)}(N_A, N_B)$. Formally,

$$\forall_{p\in\{a,b\},p'\in\{a,b\}\cup O,v\in V}\ \left[\{\mathcal{E}_{\mathcal{PK}(p)}(\mathcal{N}(\epsilon, \mathcal{I}(p)), v))\} \rightarrow \mathcal{E}_{\mathcal{PK}(p')}(v)\right]. \tag{3}$$

3 Knowledge Flow Analysis

Knowledge flow analysis is based on a simple mathematical foundation. This section formalizes the ideas outlined in the discussion of knowledge flow basics. We describe how *communication rules* direct knowledge flows (3.1), show that our treatment of primitives ensures a fixed pool of values (3.2), and formulate the analysis problem in terms of Oscar's knowledge flows (3.3).

[1]We use the parameter v in c instead of $\mathcal{N}(\epsilon, \mathcal{I}(p))$ because p', the recipient of c, cannot conclusively determine that v is, in fact, the nonce $\mathcal{N}(\epsilon, \mathcal{I}(p))$.

3.1 Communicating Knowledge

We denote the sets of all *principals* and *values* by P and V. A subset of $P \times V$ is a *state of knowledge* drawn from $K = 2^{P \times V}$, the set of all possible states of knowledge. For a given state of knowledge $k \in K$, we say that "p knows v" if $(p, v) \in k$.

Definition 1 *A tuple* (R, k_0) *is a* knowledge flow *for* (P, V) *directed by the* communication rules $R \subseteq P \times V \times P \times K$ *and originating from the state* $k_0 \in K$.

A communication rule describes the conditions under which one principal may gain knowledge from another. For example, the rule $(e, \mathcal{E}_{\mathcal{PK}(p_b)}(v), p_a, \{(p_a, \mathcal{PK}(p_b)), (p_a, v)\})$ states that the encryptor e will tell the cipher $\mathcal{E}_{\mathcal{PK}(b)}(v)$ to the principal p_a if p_a knows p_b's public key and the plaintext v.

Note that our definition of a communication rule limits the class of protocols expressible in the knowledge flow framework. In particular, our rules cannot be used to specify conditions under which information is *withheld* from a principal, such as "a will *not* tell v to b if b knows x". To the best of our knowledge, no protocol proposed for practical use requires this form of expressiveness.

Given a set of communication rules R, we say that $k' \in K$ is reachable from $k \in K$ via R if k' is the result of applying all rules in R to k at most once; i.e. $k' = f_R(k)$ where

Definition 2 $f_R : K \longrightarrow K$ *such that*

$$f_R(k) = k \cup \left\{ (p_a, v) : \begin{array}{l} (p_b, v) \in k, k_a \subseteq k, \text{ and } (p_b, v, p_a, k_a) \in R, \\ \text{for some } p_b \in P \text{ and } k_a \in K \end{array} \right\}.$$

A state of knowledge k_n is reachable in the context of a knowledge flow (R, k_0) if $k_n = f_R^n(k_0)$. The *maximal state of knowledge* $f_R^*(k_0)$ is the limit of $k_n = f_R^n(k_0)$ as $n \to \infty$. A state of knowledge $f_{R_\kappa}^*(\kappa)$ is *valid* for a knowledge flow (R, k_0) if $R_\kappa \subseteq R$ and $\kappa \subseteq k_0$. Since $f_R(k_0)$ is monotonically increasing in R and k_0, any valid state of knowledge is a subset of the maximal state of knowledge. Hence, the maximal state of knowledge is also the smallest fixed point of f_R which subsumes k_0. It is evident from Definition 2 that self-rules such as $r = (p, v, p, k_p) \in R$ do not affect the flow of knowledge: $f_R(k) = f_{R-\{r\}}(k)$. We therefore assume that R does not contain any self-rules.

3.2 Initial Knowledge

For each value v, $Source(v) = \{p : (p, v) \in k_0\}$ defines the set of principals who draw v. In the knowledge flow framework, a principal p outside of $Source(v)$ can learn v only by communicating with principals who know v. We therefore treat cryptographic primitives, and other computationally feasible algorithms, as principals. For example, suppose that, in practice, p can compute v by applying the algorithm $\mathcal{A}$ to inputs $i_1, i_2, \ldots, i_n$. We model $\mathcal{A}$ by adding the principal A to P, the tuple (A, v) to k_0, and the rule $(A, v, p, \{(p, i_1), (p, i_2), \ldots (p, i_n)\})$ to R.

Our treatment of primitives ensures that $Knowledge(k_0) = \{v : (p, v) \in k_0 \text{ for some } p \in P\}$ consists of *all* learnable values. Hence, V is the same in the initial and the maximal state of knowledge,

$$Knowledge(k_0) = Knowledge(f_R^*(k_0)), \tag{4}$$

which implies that we can safely restrict our analysis to the subset of R which only involves values in $Knowledge(k_0)$.

We further simplify our approach by constraining k_0, and therefore R, according to standard security assumptions. Specifically, we assume *the single source axiom* for values that are fixed points of cryptographic functions. For example, if the primitive h models a hashing function $\mathcal{H}$, then we assume that $\{h\} = Source(x)$ for all x such that $x = \mathcal{H}(x)$. We thus model the assumption that solving the equation $x = \mathcal{H}(x)$ is computationally hard by stating that no principal other than h can draw x:

Definition 3 (Single Source Axiom) *Set F_p is* fixed *for a principal p if for each $(p, v, p_a, k_a) \in R$ with $v \in F_p$, there exists an $x \in F_p$ such that $(p_a, x) \in k_a$. Fixed sets F_p for $p \in P$ satisfy the single-source axiom if, for all $p \in P$, $p_a \in P$, and $v \in V$, $[v \in F_p$ and $(p_a, v) \in k_0] \Rightarrow [p = p_a]$.*

The consequence of the single-source axiom is that no principal outside of $\{p\} = Source(v)$ can ever learn $v \in F_p$:

Lemma 4 *If the fixed sets F_p for $p \in P$ satisfy the single-source axiom, then, for all $p \in P$, $p_a \in P$, and $v \in V$, $[v \in F_p$ and $(p_a, v) \in f_R^n(k_0)] \Rightarrow [p = p_a$ and $(p_a, v) \in k_0]$.*

Proof. We use induction on n. The case $n = 0$ is equivalent to the single-source axiom. Suppose that the lemma holds for n, our induction hypothesis. Let $v \in F_p$ and $(p_a, v) \in f_R^{n+1}(k_0)$. According to Definition 2, (i) $(p_a, v) \in f_R^n(k_0)$ and the lemma follows from the induction hypothesis, or (ii) there exists a $p_b \in P$ and $k_a \in K$ such that $(p_b, v) \in f_R^n(k_0)$, $k_a \subseteq f_R^n(k_0)$, and $(p_b, v, p_a, k_a) \in R$. From the induction hypothesis we infer that $p = p_b$ and $(p_b, v) \in k_0$ since $v \in F_p$ and $(p_b, v) \in f_R^n(k_0)$. Hence, $(p, v, p_a, k_a) \in R$ and, since $v \in F_p$ and F_p is a fixed set for p, there exists an $x \in F_p$ such that $(p_a, x) \in k_a \subseteq f_R^n(k_0)$, which proves $p = p_a$ by the induction hypothesis. Notice that $(p_a, v) \in k_0$ because $p_a = p = p_b$ and $(p_b, v) \in k_0$. The lemma follows by induction on n.

□

Together with Equation (4), Lemma 4 implies that $f^*_{R_F}(k_0) = f^*_R(k_0) \subseteq k_0 \cup [P \times (Knowledge(k_0) - F)]$, where F is the union of all F_p and

$$R_F = \{(p_b, x, p_a, k_a) \in R : \{(p_b, x)\} \cup k_a \subseteq k_0 \cup [P \times (Knowledge(k_0) - F)]\}.$$

Hence, we need to analyze only the knowledge flows characterized by R_F.

3.3 Adversaries' Knowledge

Let $O \subseteq P$ be a group of collaborating adversaries. We collapse O into a single principal o using the following merging function:

$$\begin{array}{rcl} Merge(p) & = & \begin{cases} o \text{ if } p \in O, \\ p \text{ if } p \notin O \end{cases} \\ Merge(k) & = & \{(Merge(p), v) : (p, v) \in k\} \\ Merge(r) & = & (Merge(p_b), v, Merge(p_a), Merge(k_a)) \text{ where } r = (p_b, v, p_a, k_a) \in R \end{array}$$

The merging of adversaries does not rule out any attacks because $Merge(f^*_R(k_0)) \subseteq f^*_{Merge(R)}(Merge(k_0))$. We subsequently assume that $Merge$ is implied and use P, R, and k_0 to refer to $Merge(P)$, $Merge(R)$, and $Merge(k_0)$.

Security properties of protocols are expressed as predicates on the values known to Oscar in the maximal state of knowledge. We therefore focus our analysis of knowledge flows to finding all the values in the projection of $f^*_{R_F}(k_0)$ on Oscar. Specifically, we introduce the projection function g and show that its smallest fixed point is the image of Oscar under $f^*_{R_F}(k_0)$.

Definition 5 *Let $X \to x$ or, more explicitly, $X \to_p x$ denote the existence of a rule $(p, x, o, k_\sigma) \in R_F$ for some $p \in P - \{o\}$ and $k_\sigma \in K$ with $X = \{v : (o, v) \in k_\sigma\}$. We define $g : 2^V \longrightarrow 2^V$ as*

$$g(X) = X \cup \{x : X_\sigma \to x \text{ for some } X_\sigma \subseteq X\}.$$

The set of values reachable from X is given by $g^*(X)$, which is the limit of $g^n(X)$ as $n \to \infty$.

Since $f_R(k_0)$ is monotonically increasing in R, Oscar's pool of values under $f_R^*(k_0)$ is maximized if (a) Oscar tells everything he knows to the honest principals and (b) the honest principals tell each other all values learnable in polynomial time–which, in our framework, are the values in $Knowledge(k_0) - F$. Formally, Oscar's final knowledge is maximized when $[(P - \{o\}) \times (Knowledge(k_0) - F)] \subseteq f_R^*(k_0)$. This is equivalent to assuming that $[(P - \{o\}) \times (Knowledge(k_0) - F)] \subseteq k_0$ because $k \subseteq f_R^*(k_0)$ implies that $f_R^*(k_0) = f_R^*(k_0 \cup k)$.

Theorem 6 (Knowledge Flow Analysis) *Let $[(P - \{o\}) \times (Knowledge(k_0) - F)] \subseteq k_0$ and let $k_n = f_R^n(k_0)$. Then the set $X_n = \{v : (o, v) \in k_n\}$ has the property that $X_n = g^n(X_0)$.*

Proof. We use induction on n. For $n = 0$, $X_n = X_0 = g^n(X_0)$. Let $X_n = g^n(X_0)$, our induction hypothesis. We now need to prove that $X_{n+1} = g^{n+1}(X_0)$. By the definition of X_{n+1}, $x \in X_{n+1} \iff (o, x) \in k_{n+1} = f_R(k_n)$. According to Definition 2, $(o, x) \in f_R(k_n)$ if and only if (i) $(o, x) \in k_n$, which is equivalent to $x \in X_n$, or (ii) there exists a $p \in P$ and $k_\sigma \in K$ such that $(p, x) \in k_n$, $k_\sigma \subseteq k_n$, and $(p, x, o, k_\sigma) \in R$. Since there are no self-rules $(o, v, o, k_\sigma) \in R$, we know that $p \in P - \{o\}$. Since $[(P - \{o\}) \times (Knowledge(k_0) - F)] \subseteq k_0 \subseteq k_n$, $(p, x) \in k_n$ if and only if $(p, x) \in k_0 \cup [P \times (Knowledge(k_0) - F)]$ by Lemma 4. This argument also proves that the condition $k_\sigma \subseteq k_n$ is equivalent to

$$X_\sigma = \{v : (o, v) \in k_\sigma\} \subseteq \{v : (o, v) \in k_n\} = X_n \text{ and } k_\sigma \subseteq k_0 \cup [P \times (Knowledge(k_0) - F)].$$

Notice that $\{(p, x)\} \cup k_\sigma \subseteq k_0 \cup [P \times (Knowledge(k_0) - F)]$ gives us $(p, x, o, k_\sigma) \in R_F$. Therefore, case ii) holds if and only if there exists a set $X_\sigma \subseteq X_n$ such that $X_\sigma \to x$. By Definition 5, case (i) or case (ii) holds if and only if $x \in g(X_n)$. Hence, $X_{n+1} = g(X_n)$ and the theorem follows by induction on n.

□

4 Two-Phase Theory

The formalism developed in the previous sections enables a systematic and efficient analysis of Oscar's knowledge flows. Specifically, Oscar's final pool of values can be computed in *two phases* by first applying all the *decomposing* rules in R and then all the *composing* ones. This is a consequence of the 'two-phase theory' [15, 10], which we now formulate and prove in the knowledge flow framework.

Intuitively, a *composing* rule combines its inputs into an output value from which some or all of the inputs can be extracted using a corresponding *decomposing* rule, if one exists. For example, the composing rule $r_z = \{x, y\} \to_p x + iy$, where $x \neq 0$ and $y \neq 0$, combines the non-zero real numbers x and y into the complex number $x + iy$. The corresponding decomposing rules, $r_x = \{x + iy\} \to_p x$ and $r_y = \{x + iy\} \to_p y$, reconstruct the inputs to r_z from its output. Formally, we define composing and decomposing rules as follows:

Definition 7 *Let p be a principal with a partial ordering $\prec_p$ on the set of values V. We call $X \to_p x$*

- *composing, if $X \prec_p x$, that is, $v \prec_p x$ for all $v \in X$, and*
- *decomposing, if there exists a value $v \in X$ with $x \prec_p v$ such that $x \in X'$ for all composing $X' \to_p v$. We say that v controls$_p$ x.*

Principal p is composing/decomposing if there exists a partial ordering $\prec_p$ such that for all $X \subseteq V$ and $x \in V$, $X \to_p x$ is composing or decomposing.

Our definition permits the images of composing rules of different principals to intersect. In practice, however, such intersections are hard to compute; that is, equations like $\mathcal{H}(z) = \mathcal{E}_k(x)$, where $\mathcal{H}$ is a hashing and $\mathcal{E}$ an encryption function, cannot be solved in polynomial time. We model this by assuming that the images of different principals' composing rules are disjoint:

Definition 8 (Global Collision Free Axiom) *Orderings $\prec_p$ are globally collision free if the sets $\{v : \exists_x [x \prec_p v]\}$ have empty intersections for different p.*

By Definition 7, the set $\{v : \exists_x [x \prec_p v]\}$ is the image of the composing rules of a composing/decomposing principal p. Hence, the global collision free axiom gives us the required condition that $[X \rightarrow_p v$ is composing and $X' \rightarrow_{p'} v$ is composing$] \Rightarrow [p = p']$ for all p and p' in P, X and X' in V, and $v \in V$.

The two phase theory (Theorem 9) follows immediately from Definitions 7 and 8. It states that applying a decomposing rule after a corresponding composing rule yields no new information. We can therefore derive Oscar's maximal state of knowledge in a minimal number of steps by applying all the decomposing rules before their composing counterparts.

Theorem 9 (Two-Phase Theory) *Suppose that the orderings of principals in $P - \{o\}$ are globally collision free: If $X \rightarrow x$ is decomposing, v controls x, $v \in X$, $X' \neq \emptyset$, and $X' \rightarrow v$ is composing, then $x \in X'$.*

Proof. Suppose that v controls$_p$ x, $X' \neq \emptyset$, and $X' \rightarrow_{p'} v$ is composing. Since v controls$_p$ x, $x \prec_p v$ and since $X' \rightarrow_{p'} v$ is composing, there exists a value x' such that $x' \prec_{p'} v$. The orderings $\prec_p$ and $\prec_{p'}$ are globally collision free, therefore $p' = p$. By the definition of v controls$_p$ x, $x \in X'$ because $X' \rightarrow_p v$ is composing.

□

5 Composing/Decomposing Principals

The applicability of the two-phase theory is not restricted by its formulation in terms of composing/decomposing principals. This section presents a general criterion for identifying composing/decomposing principals which we use in the next section to demonstrate that both standard cryptographic primitives and protocol rules are composing/decomposing in our framework.

We represent composing and decomposing rules with *locally collision free sets*. This representation ensures that each decomposing rule has a corresponding composing rule (5) and that the composing rules are locally free of collisions (6)—i.e., for all p in P, all X and X' subsets of V, and all $v \in V$, $[X \rightarrow_p v$ is composing and $X' \rightarrow_p v$ is composing$] \Rightarrow [X = X']$.

Definition 10 (Local Collision Free Axiom) *A set $S \subseteq V^m$ is locally collision free if there exist sets C and D, which are subsets of $\{1, \ldots, m\}$, such that there exist subsets $W_i \subseteq \{1, \ldots, m\}$, $i \in C \cup D$, with the following properties:*

$$\text{for all } i \in D \text{ there exists a } h \in C \text{ such that } h \in W_i \text{ and } i \in W_h \tag{5}$$

and

$$\begin{aligned} &\text{for all } i, t \in C \text{ and for all } (x_1, \ldots, x_m), (y_1, \ldots, y_m) \in S, \\ &\text{if } x_i = y_t \text{ then } \{x_j : j \in W_i\} = \{y_j : j \in W_t\}. \end{aligned} \tag{6}$$

The image $Im(S)$ of S is defined as the set of values x_i for some $(x_1, \ldots, x_i, \ldots x_m) \in S$ such that $i \in C$.

An example of a locally collision free set is

$$S = \{(s, \mathcal{G}(s), x, \mathcal{E}_{\mathcal{G}(s)}(x), \mathcal{S}_s(x)) : s, x \in V\} \subseteq V^5,$$

where $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{S}$ are injective functions free of collisions; that is, $\mathcal{G}(v) \neq \mathcal{E}_{\mathcal{G}(s)}(x)$, $\mathcal{G}(v) \neq \mathcal{S}_s(x)$, and $\mathcal{E}_{\mathcal{G}(s)}(x) \neq \mathcal{S}_v(y)$ for all s, v, x, and y in V. Let $D = \{3\}$ indicate the position in the tuples in S which correspond to x and let $C = \{2, 4, 5\}$ indicate the positions which correspond to $\mathcal{G}(s)$, $\mathcal{E}_{\mathcal{G}(s)}(x)$, and $\mathcal{S}_s(x)$. Let $W_3 = \{1, 4\}$, $W_2 = \{1\}$, $W_4 = \{2, 3\}$, and $W_5 = \{1, 3\}$. Since $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{S}$ are injective functions with disjoint images, S satisfies (6). Condition (5) is satisfied by taking $h = 4 \in C$ and $i = 3 \in D$.

The following theorem shows how a local collision free set leads to a composing/decomposing principal. Its proof is in Appendix A.

Theorem 11 (Composing/Decomposing) *Let $p \in P - \{o\}$ be a principal such that*

$$\begin{aligned}(p, v, p_a, k_a) \in R \textit{ implies}& \\ &\textit{there exists an } i \in C \cup D \textit{ and } (x_1, \ldots, x_m) \in S \textit{ such that} \\ &v = x_i, k_a = \{(p_a, x_j) : j \in W_i\},\end{aligned} \tag{7}$$

where S is local collision free with respect to C, D, and W_i, $i \in C \cup D$. Let F_p be the maximal[2] fixed set with $F_p \subseteq Im(S)$. Then, p is composing/decomposing; composing rules correspond to $i \in C$ and decomposing rules correspond to $i \in D$. The image of the composing rules is $\{v : \exists_{x \in V}\ [x \prec_p v]\} \subseteq Im(S)$.

Applying Theorem 11 to our example, we define the encryptor/decryptor/signer e by the decomposing rule $(e, x, p, \{(p, s), (p, \mathcal{E}_{\mathcal{G}(s)}(x))\})$ and the composing rules $(e, \mathcal{G}(s), p, \{(p, s)\})$, $(e, \mathcal{E}_{\mathcal{G}(s)}(x), p, \{(p, \mathcal{G}(s)), (p, x)\})$ and $(e, \mathcal{S}_s(x), p, \{(p, s), (p, x)\})$. The decomposing rule corresponds to the position $3 \in D$ and models decryption; the composing rules correspond to the positions $2 \in C$, $4 \in C$, and $5 \in C$ and model public key generation, encryption, and signing. The principal e is therefore composing/decomposing, and the two-phase theory holds for $\mathcal{E}$, $\mathcal{G}$, and $\mathcal{S}$.

The composing/decomposing theorem is compatible with the knowledge flow theorem if the fixed set F_p satisfies the single source axiom. In Appendix A, we show that this is equivalent to assuming that it is hard to solve the equation $x = w(x)$ where $w(x) = \mathcal{S}_a(\mathcal{E}_b(\mathcal{G}(\mathcal{E}_{\mathcal{G}(\mathcal{S}_c(x))}(d))))$ is some function composed of $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{S}$.

6 Primitives

We now present a sample library of composing/decomposing primitives, which is sufficient for modeling a wide range of security protocols. The library includes the standard cryptographic primitives: encryption/decryption, signing, pair/set construction, nonce generation, and hashing. It also provides a special *rule primitive* that allows protocol rules to be modeled in the composing/decomposing pattern.

6.1 Cryptographic Primitives

Encryption/Decryption Public key encryption [31] consists of a (probabilistic) encryption algorithm, a decryption algorithm, and a (probabilistic) key-generating algorithm. Given some security parameter, the key-generating algorithm generates a public-secret key pair. We model p's private key $\mathcal{SK}(p)$ as belonging to p's initial knowledge, $(p, \mathcal{SK}(p)) \in k_0$. We model the public key as a one-way function of the secret key, i.e., $\mathcal{PK}(p) = \mathcal{G}(\mathcal{SK}(p))$. Hence, one can compute a corresponding public key from the given secret

[2]There exists a unique maximal fixed set since the union of two fixed sets is again a fixed set.

key but not vice versa. Letting e denote the principal representing public key encryption, we can express key-generation as follows: for all $p \in P$ and $s \in V$, $(e, \mathcal{G}(s), p, \{(p, s)\}) \in R$. If we project this family of rules on Oscar, we obtain

$$\forall_{s \in V}\ [\{s\} \rightarrow \mathcal{G}(s)]. \tag{8}$$

Given a plaintext x and a public key $\mathcal{G}(s)$, the encryption algorithm computes the ciphertext[3] $\mathcal{E}_{\mathcal{G}(s)}(x)$. Thus, the parameterized rule for encryption is, for all $p \in P$ and $s, x \in V$, $(e, \mathcal{E}_{\mathcal{G}(s)}(x), p, \{(p, \mathcal{G}(s)), (p, x)\}) \in R$. Projecting the rule on Oscar yields

$$\forall_{s,x \in V}\ [\{x, \mathcal{G}(s)\} \rightarrow \mathcal{E}_{\mathcal{G}(s)}(x)]. \tag{9}$$

Given a ciphertext $\mathcal{E}_{\mathcal{G}(s)}(x)$ and the secret key s, the decryption algorithm computes the plaintext x. Hence, for all $p \in P$ and $s, x \in V$, $(e, x, p, \{(p, s), (p, \mathcal{E}_{\mathcal{G}(s)}(x))\}) \in R$ and

$$\forall_{s,x \in V}\ [\{s, \mathcal{E}_{\mathcal{G}(s)}(x)\} \rightarrow x]. \tag{10}$$

Signing We can model digital signatures [31] by extending e with the rules of the form $(e, \mathcal{S}_s(x), p, \{(p, s), (p, x)\}) \in R$, for all $p \in P$ and $s, x \in V$, which translate into

$$\forall_{v \in V}\ [\{x, s\} \rightarrow \mathcal{S}_s(x)]. \tag{11}$$

In practice, the principal who receives x and its signature $S_s(x)$ can verify the signature by using the public key $\mathcal{G}(s)$. In our model, it is sufficient to note that knowledge of $y = \mathcal{S}_s(x)$ already verifies that y is a signature of x, signed by using the secret key s. That is, the principal who obtained y from e knows both s and x.

Symmetric key encryption is modeled by (9) and (10) where $G(s)$ is replaced by s and where s represents the symmetric key. We can extend this definition with (11) to include message authentication codes (MACs).

Pairing/Set Construction Let t be a principal such that a communication rule (t, v, p_a, k_a) is in R if and only if one of the following holds for $v \in V$, $p_a \in P$, and $k_a \in K$: (i) $v = (x, y)$ and $k_a = \{(p_a, x), (p_a, y)\}$ for some $x, y \in V$, (ii) $k_a = \{(p_a, (v, y))\}$ for some $y \in V$, or (iii) $k_a = \{(p_a, (x, v))\}$ for some $x \in V$. Projected on Oscar, this set of rules becomes

$$\forall_{x,y \in V}\ [\{x, y\} \rightarrow (x, y)], \tag{12}$$

$$\forall_{x,y \in V}\ [\{(x, y)\} \rightarrow x], \tag{13}$$

$$\forall_{x,y \in V}\ [\{(x, y)\} \rightarrow y]. \tag{14}$$

Replacing (x, y) by $\{x, y\}$ in (12-14) turns t into a primitive that generates sets of cardinality 2.

Nonce Generation Let $\mathcal{I}(p) \in V$ represent the public identity of p (the identity function $\mathcal{I}$ embeds P in V). We model nonce generation with the nonce primitive n and the parameterized rule $\forall_{p \in P, v \in V}(n, \mathcal{N}(v, \mathcal{I}(p)), p, \{(p, v)\}) \in R$, which translates into

$$\forall_{p \in O, v \in V}\ [\{v\} \rightarrow \mathcal{N}(v, \mathcal{I}(p))]. \tag{15}$$

The dependence of n's output on $\mathcal{I}(p)$ ensures that p cannot learn other principals' nonces from n. The parameter v represents the seed from which a pseudo random nonce is generated. If a protocol stipulates that a principal p needs to generate a new nonce in response to a received message m, then v is taken to be equal to m. For the first nonce of a protocol, we take v to be the empty string ϵ.

Hashing We define the primitive h for calculating hashes $\mathcal{H}(x)$ with the family of rules $\forall_{p \in P, x \in V}\ (h, \mathcal{H}(x), p, \{(p, x)\}) \in R$ and

$$\forall_{x \in V}\ [\{x\} \rightarrow \mathcal{H}(x)]. \tag{16}$$

[3] If the algorithm is probabilistic (for example in ElGamal encryption) then the ciphertext $\mathcal{E}_{\mathcal{G}(s)}(x; r)$ is also a function of some random value r (uniformly) drawn by the algorithm.

6.2 The Rule Primitive

Protocol rules do not compute new values; rather, they model the transmission of values computed by the primitives. We can therefore embed protocol rules into the initial state of knowledge as follows.

With each (parameterized) protocol rule $x \to y$, we associate the value $|x \to y| \in V$. This value is composed/decomposed by the *rule primitive* r via the parameterized rules

$$\forall_{p\in P,x,y\in V}\ (r, |x \to y|, p, \{(p,x),(p,y)\}) \in R \text{ and}$$

$$\forall_{p\in P,x,y\in V}\ (r, y, p, \{(p,|x \to y|),(p,x)\}) \in R,$$

whose projected forms are

$$\forall_{x,y\in V}\ [\{x,y\} \to |x \to y|] \text{ and} \tag{17}$$

$$\forall_{x,y\in V}\ [\{x,|x \to y|\} \to y]. \tag{18}$$

We represent each protocol rule $x \to y$ with the initial knowledge $(o, |x \to y|)$. Oscar can now use (18) to learn y from $|x \to y|$ if he knows x. For example, the following addition to k_0, together with (18), simulates the rule (3) of the Needham-Schroeder protocol:

$$\forall_{p\in\{a,b\},p'\in\{a,b\}\cup O,v\in V}\left[(o, |\{\mathcal{E}_{\mathcal{PK}(p)}(\mathcal{N}(\epsilon,\mathcal{I}(p)),v))\} \to \mathcal{E}_{\mathcal{PK}(p')}(v)|) \in k_0\right].$$

6.3 Summary

The rules (8)-(18) define a library of primitives—e, t, n, h and r—that are composing/decomposing according to Theorem 11. The following assumptions are implicit in their definitions:

The Collision Free Axioms It is hard to compute collisions of the composition rules (8), (9), (11), (12), (15), (16) and (17). Therefore, we model these rules as injective functions that are mutually free of collisions; that is, they satisfy the local and global collision free axioms.

The Single Source Axiom It is hard to compute fixed points of functional compositions of the rules (8), (9), (11), (12), (15), (16) and (17). The principals e, t, n, h and r hence satisfy the single-source axiom.

Cryptographic Primitive Properties It is hard to compute the inverses that are not encoded by the decomposition rules (10), (13), and (14). The rules (8-14) represent Oscar's computational means in The Dolev-Yao intruder model [14]. This assumes perfect cryptography: the set of values is supposed to be a free algebra.

Collision freeness and perfect cryptography are routinely assumed when reasoning about security protocols. To the best of our knowledge, however, the necessity of assuming the single source axiom has not been recognized before. We discovered it using the Alloy Analyzer [23], a general purpose model finder, to check a security theorem about knowledge flows in the CPUFs renewal protocol [19, 18]: in the absence of the axiom, the Analyzer generates a false counterexample to the theorem based on a fixed value that satisfies the equation $x = \mathcal{E}_s(x)$.

7 Related Work

The first formalisms designed for reasoning about cryptographic protocols are belief logics such as BAN logic [8], used by the Convince tool [25] with the HOL theorem prover [22], and its generalizations (GNY

[21], AT [3], and SVO logic [41] which the C3PO tool [13] employs with the Isabelle theorem prover [37]). Belief logics are difficult to use since the logical form of a protocol does not correspond to the protocol itself in an obvious way. Almost indistinguishable formulations of the same problem lead to different results. It is also hard to know if a formulation is over constrained or if any important assumptions are missing. BAN logic and its derivatives cannot deal with security flaws resulting from interleaving of protocol steps [7] and cannot express any properties of protocols other than authentication [28]. To overcome these limitations, the knowledge flow formalism has, like other approaches [27, 33, 11, 40, 30], a concrete operational model of protocol execution. Our model also includes a description of how the honest participants in the protocol behave and a description of how an adversary can interfere with the execution of the protocol.

Specialized model checkers such as Casper [27], Murϕ [33], Brutus [11], TAPS [12], and ProVerif [1] have been successfully used to analyze security protocols. These tools are based on state space exploration which leads to an exponential complexity. Athena [40] is based on a modification of the strand space model [16]. Even though it reduces the state space explosion problem, it remains exponential. Multiset rewriting [15] in combination with tree automata is used in Timbuk [17]. The relation between multiset rewriting and strand spaces is analyzed in [9]. The relation between multiset rewriting and process algebras [32, 2] is analyzed in [5].

Proof building tools such as NRL, based on Prolog [30], have also been helpful for analyzing security protocols. However, they are not fully automatic and often require extensive user intervention. Model checkers lead to completely automated tools which generate counterexamples if a protocol is flawed. For theorem-proving-based approaches, counterexamples are hard to produce.

For completeness, we note that if the initial knowledge of the intruder consists of a finite number of explicit (non-parameterized, non-symbolic) values, then a polynomial time intruder detection algorithm can be shown to exist using a generalization of the proof normalization arguments [29, 4, 20], which were employed in [6, 35] and have been implemented in the framework [36] (our two phase theorem can also be used to derive a polynomial time algorithm). However, in practice, the initial knowledge of an intruder is unbounded and represented by a finite number of parameterized sets, each having an infinite number of elements.

8 Concluding Remarks

We introduced knowledge flow analysis, a new framework for reasoning about knowledge in cryptographic protocols. The key advantage of the knowledge flow approach over other formalisms is its simplicity and flexibility. It is simple in the sense that the underlying mathematics is straightforward and elementary; it does not require any specialized background (in logic). It is flexible in the sense that the same library of cryptographic primitives can be used to model different protocols and that the security of a complex scheme involving multiple protocols can be verified. Knowledge flow analysis allows modeling of confidentiality and authenticity via a wide range of primitives such as pairing, union, hashing, symmetric key encryption, public key encryption, MACs and digital signatures.

Our formalism derives its simplicity from being just sufficiently expressive to enable modeling of practical cryptographic protocols. In particular, existentials [15] cannot be encoded as knowledge flows; existentials are implicitly modeled in Oscar's initial knowledge. NP-hardness proofs which use (existential) Horn clause reduction [15] or SAT3 reduction [39] are not applicable to knowledge flow analysis.

Our formalism leads to a rigorous mathematical treatment and generalization of the two-phase theory [15, 10] which is used to efficiently verify protocols. Our treatment reveals the necessary and sufficient collision free and single source axioms; it is hard to compute collisions and fixed points of compositions of cryptographic primitives. To the best of our knowledge the necessity of assuming the single source axiom has not been recognized before.

A Fixed Sets and Orderings

To prove Theorem 11, we define a sequence of subsets which we use to define a partial ordering and to characterize a fixed set.

Definition 12 *Let S be locally collision free with respect to C, D, and W_i, $i \in C \cup D$. We define $S_n \subseteq V$ recursively by $S_{-1} = \emptyset$,*

$$S_0 = V - Im(S) = V - \{x_i : i \in C \text{ and there exists a tuple } (x_1, \ldots, x_m) \in S\},$$

and, for $n \geq 0$,

$$S_{n+1} = S_n \cup \left\{ x_i : \begin{array}{l} i \in C \text{ and there exists a } (x_1, \ldots, x_m) \in S \\ \text{such that } x_j \in S_n \text{ for } j \in W_i \end{array} \right\}.$$

We define $S_\infty = \{v \in V : v \in S_n \text{ for some } n \geq 0\}$.

We first show in Lemma 14 that $V - S_\infty$ is a fixed set. We start with a result which we use throughout the whole proof.

Lemma 13 *Let $i \in C$, $(x_1, \ldots, x_m) \in S$, and $x_i \in S_\infty$. Then (i) $x_i \in S_{n+1} - S_n$ for some $n \geq 0$ and (ii) $x_j \in S_n$ for all $j \in W_i$ and there exists a $h \in W_i$ such that $x_h \in S_n - S_{n-1}$.*

Proof. (i) Since $i \in C$, $x_i \notin S_0$ which proves $n \geq 0$. (ii) Let $k \geq 0$ be the smallest index for which $x_j \in S_k$ for all $j \in W_i$. Then, there exists an index $h \in W_i$ such that $x_h \in S_k - S_{k-1}$. Notice that $x_i \in S_{k+1}$ by the definition of S_{k+1}. Therefore, if $k < n$ then $x_i \in S_{k+1} \subseteq S_n$, contradicting $x_i \in S_{n+1} - S_n$. This proves $n \leq k$.

From the definition of $x_i \in S_{n+1} - S_n$ we infer that there exists a $t \in C$, $(y_1, \ldots, y_m) \in S$ such that $x_i = y_t$ and $y_j \in S_n$ for $j \in W_t$. Since $i \in C$, $t \in C$, and $x_i = y_t$, (6) yields $x_h \in \{x_j : j \in W_i\} = \{y_j : j \in W_t\} \subseteq S_n$. From $x_h \in S_k - S_{k-1}$, we infer $n \geq k$. We conclude $n = k$ which proves the lemma.

□

Lemma 14 *$V - S_\infty$ is a fixed set for p.*

Proof. Let $(p, v, p_a, k_a) \in R$ with $v \in V - S_\infty$. From (7) we infer that there exists an $i \in C \cup D$ and $(x_1, \ldots, x_m) \in S$ such that $v = x_i$ and $k_a = \{(p_a, x_j) : j \in W_i\}$. If $i \in C$ and $x_j \in S_\infty$ for all $j \in W_i$, then, by the definition of S_∞, $x_i \in S_\infty$, which contradicts $x_i = v \in V - S_\infty$. Hence, if $i \in C$ then $x_j \in V - S_\infty$ for some $j \in W_i$.

If $i \in D$, then (5) shows the existence of a $h \in C$ with $i \in W_h$ and $h \in W_i$. From $i \in W_h$ and $h \in C$ we infer that if $x_h \in S_\infty$ then, by Lemma 13 (ii), $x_i \in S_\infty$, which contradicts $x_i = v \in V - S_\infty$. Hence, $x_j \in V - S_\infty$ for some $j \in W_i$.

□

Let F_p be the maximal fixed set such that $F_p \subseteq V - S_0 = Im(S)$. Then Lemma 14 proves that $V - S_\infty \subseteq F_p$, hence, $V - F_p \subseteq S_\infty$. Notice that $X \rightarrow_p x$ (defined by using R_F) implies that there exists a rule $(p, x, o, k_\sigma) \in R$ such that $X = \{v : (o, v) \in k_\sigma\}$ and $X \cap F_p = \emptyset$. Since F_p is a fixed set, $x \in F_p$ contradicts $X \cap F_p = \emptyset$. Thus, for $X \rightarrow_p x$, both $x \notin F_p$ and $X \cap F_p = \emptyset$, that is $x \in S_\infty$ and $X \subseteq S_\infty$.

Sets S_n lead to the partial ordering

$$[v \prec_p w] \quad \equiv \quad [v \in S_a - S_{a-1} \text{ and } w \in S_b - S_{b-1} \text{ for some } 0 \leq a < b].$$

Notice that $\{v : \exists_{x \in V} [x \prec_p v]\} \subseteq V - S_0 = Im(S)$. Theorem 11 follows from Lemmas 15-16, which prove that p is composing/decomposing.

Lemma 15 *Let $(x_1, \ldots, x_m) \in S$ with $\{x_j : j \in W_i\} \rightarrow_p x_i$. Then, $\{x_j : j \in W_i\} \rightarrow_p x_i$ is composing if and only if $i \in C$.*

Proof. Suppose that $i \in C$. Since $\{x_j : j \in W_i\} \rightarrow_p x_i$, $x_i \in S_\infty$. By Lemma 13 (i), $x_i \in S_{n+1} - S_n$ for some $n \geq 0$, by Lemma 13 (ii), $\{x_j : j \in W_i\} \prec_p x_i$, which proves that $\{x_j : j \in W_i\} \rightarrow_p x_i$ is composing.

Suppose that $\{x_j : j \in W_i\} \rightarrow_p x_i$ is composing, that is, $\{x_j : j \in W_i\} \prec_p x_i$. If $i \in D$, then by (5) there exists a $h \in C$ with $h \in W_i$ and $i \in W_h$. Since $\{x_j : j \in W_i\} \rightarrow_p x_i$, $x_h \in S_\infty$ and by Lemma 13 (i), $x_h \in S_{n+1} - S_n$ for some $n \geq 0$. By Lemma 13 (ii), $x_i \in \{x_j : j \in W_h\} \subseteq S_n$, hence, $x_i \prec_p x_h$. This contradicts $\{x_j : j \in W_i\} \prec_p x_i$ and we conclude that $i \notin D$, that is, $i \in C$.

□

Lemma 16 *Let $(x_1, \ldots, x_m) \in S$ with $\{x_j : j \in W_i\} \rightarrow_p x_i$. Then, $\{x_j : j \in W_i\} \rightarrow_p x_i$ is decomposing if and only if $i \in D$.*

Proof. We prove $\{x_j : j \in W_i\} \rightarrow_p x_i$ is decomposing for $i \in D$. Then, the lemma follows from Lemma 15 since $i \notin D \Longleftrightarrow i \in C$ lead to composing rules. By (5), there exists a $h \in C$ with $h \in W_i$ and $i \in W_h$. By Lemma 13 (i), $x_h \in S_{n+1} - S_n$ for some $n \geq 0$. By Lemma 13 (ii) $x_i \in S_n$, hence, $x_i \prec_p x_h$.

To prove the lemma we show that x_h controls$_p$ x_i. Let $\{y_j : j \in W_t\} \rightarrow_p y_t = x_h$ be composing, hence, $t \in C$ by Lemma 15. Then (6) with $h \in C$ and $x_h = y_t$ states $\{x_j : j \in W_h\} = \{y_j : j \in W_t\}$. This proves $x_i \in \{y_j : j \in W_t\}$ and we conclude that x_h controls$_p$ x_i.

□

The composing/decomposing theorem is compatible with the knowledge flow theorem if F_p satisfies the single source axiom. We need to show that it is hard to compute an element $v_0 \in F_p$. First, observe that locally collision free sets often satisfy the following condition that is slightly stronger than (6): for each $i \in C$ there exists an injective *function* c_i such that $c_i((x_j)_{j \in W_i}) = x_i$ for $(x_1, \ldots, x_m) \in S$ and such that the image of c_i has an empty intersection with the images of c_j, $j \neq i$; $Im(S)$ is equal to the union of the images of c_i, $i \in C$.

Since $F_p \subseteq Im(S)$, $v_0 = c_i(x_1, \ldots, x_m)$ for some $i \in C$. Since F_p is a fixed set, $v_1 \in F_p$ for some $v_1 = x_j$. Thus, $v_0 = q_0(v_1)$ for some function derived from c_i. By continuing this argument we obtain a sequence of elements $v_0 = q_0(v_1), v_1 = q_1(v_2), \ldots$ In practise, the domain of the functions c_i, $i \in C$, has a finite size. Thus there exist $j < h$ with $v_j = v_h$, that is, v_j is a fixed point of the equation $x = w(x)$ where $w(x) = q_j(q_{j+1}(\ldots q_{h-1}(x)\ldots))$ is some function composed of c_i, $i \in C$. So, the single source axiom is satisfied if it is hard to compute compositions $q_0(q_1(\ldots q_{j-1}(x)\ldots))$ with $x = w(x)$.

References

[1] M. Ababi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1):102–146, 2005.

[2] M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *Proc. of CONCUR '97: Concurrency Theory, 8th International Conference, LNCS 1243*, pages 59–73, 1997.

[3] M. Abadi and M.R. Tuttle. A semantics for a logic of authentication. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, 1991.

[4] D. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. *JACM*, 48(1):70–109, 2001.

[5] S. Bistarelli, I. Cervesato, G. Lenzini, and F. Martinelli. Relating process algebras and multiset rewriting for immediate decryption protocols. In *2nd Int. Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS), LNCS 2776*, pages 86–99, 2003.

[6] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Flow logic for dolev-yao secrecy in cryptographic processes. *Future Gener. Comput. Syst.*, 18(6):747–756, 2002.

[7] C. Boyd and W. Mao. On a limitation of ban logic. In *Advances in Cryptology: Eurocrypt '93, Springer-Verlag*, pages 240–247, 1993.

[8] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[9] I. Cervesato, N. Durgin, J.Mitchell, P. Lincoln, and A. Scedrov. A comparison between strand spaces and multiset rewriting for security protocol analysis. In *Software Security Theories and Systems, Mext-NSF-JSPS International Symposium, ISSS 2002, LLNCS 426*, 2003.

[10] E. M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings of the IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*, 1998.

[11] E.M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with brutus. *ACM Transactions on Software Engineering and Methodology*, 9(4):443–487, 2000.

[12] E. Cohen. TAPS: A first-order verifier for cryptographic protocols. In *Computer Security Foundations Workshop*, 2004.

[13] Anthony H. Dekker. C3po: A tool for automatic sound cryptographic protocol analysis. In *13th IEEE Computer Security Foundations Workshop (CSFW'00)*, 2000.

[14] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[15] N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 1:677–722, 2004.

[16] F. Javier Thayer Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, 1998.

[17] G. Feuillade, T. Genet, and V. Viet Triem Tong. Reachability analysis of term rewriting systems. *Technical Report RR-4970, INRIA, 2003, to be published in the Journal of Automated Reasoning*, 2004.

[18] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *Proceedings of the 18th Annual Computer Security Applications Conference*, 2002.

[19] Blaise L. P. Gassend. Physical random functions. Master's thesis, MIT, 2003.

[20] Robert Givan and David Mcallester. Polynomial-time computation via local inference relations. *ACM Trans. Comput. Logic*, 3(4):521–541, 2002.

[21] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 234–248, 1990.

[22] M. J. C. Gordon and T. F. Melham. *Introduction to HOL, a theorem proving environment for higher-order logic*. Cambridge University Press, Cambridge, England, 1993.

[23] Daniel Jackson. Automating first-order relational logic. In *Proc. ACM SIGSOFT Conf. Foundations of Software Engineering / European Software Engineering Conference (FSE/ESEC '00)*, 2000.

[24] Daniel Jackson. Alloy: a lightweight object modelling notation. *ACM TOSEM*, 11(2):256–290, 2002.

[25] Randall W. Lichota, Grace L. Hammonds, and Stephen H. Brackin. Verifying cryptographic protocols for electronic commerce. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 53–65, 1996.

[26] G. Lowe. Breaking and fixing the needham-schröder public-key protocol using csp and fdr. In *2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, 1996.

[27] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proceedings of the 1997 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 18–30, 1997.

[28] W. Mao and C. Boyd. Towards formal analysis of security protocols. In *Proceedings of the Computer Security Foundation Workshop VI*, pages 147–158, 1993.

[29] David McAllester. Automatic recognition of tractability in inference relations. *Journal of ACM*, 40(2), 1993.

[30] Catherine A. Meadows. The nrl protocol analyzer: An overview. In *Proceedings of the 2nd Conference on the Practical Applications of Prolog*, 1994.

[31] A.J. Menezes, P.C. van Oorschor, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[32] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 2000.

[33] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murϕ. In *Proceedings of the 1997 IEEE Symposium on Research in Security and Privacy*, pages 141–153, 1997.

[34] R. Needham and M. Schröder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[35] Flemming Nielson, Hanne Riis Nielson, and Helmut Seidl. Cryptographic analysis in cubic time. In *TOSCA'01*, volume 62 of *ENTCS*, 2001.

[36] Flemming Nielson, Hanne Riis Nielson, Hongyan Sun, Mikael Buchholtz, Rene Rydhof Hansen, Henrik Pilegaard, and Helmut Seidl. The succinct solver suite. In *10th TACAS*, volume 2988 of *LNCS*, 2004.

[37] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL Tutorial Draft*, March 8 2002.

[38] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21:8–10, January 1987.

[39] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Proceedings of the 14th Computer Security Foundations Workshop*, pages 174–187, 2001.

[40] D. Song, S. Berezin, and A. Perrig. Athena, a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1), 2001.

[41] P.F. Syverson and P.C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of the 13th Symposium on Security and Privacy*, 1994.