Distinguishability in Protocol Analysis: Formally Analyzing Guessing Attacks

Siva Anantharaman¹ and Kimberly A. Gero² and Paliath Narendran³ and Michael Rusinowitch⁴

¹ LIFO - Université d’Orleans (France), e-mail: siva@univ-orleans.fr
² The College of Saint Rose (USA), e-mail: gerok@strose.edu
³ University at Albany–SUNY (USA), e-mail: pnarendran@albany.edu
⁴ Loria-INRIA Lorraine, Nancy (France), e-mail: rusi@loria.fr

Abstract—Symbolic cryptographic protocol analysis is an important topic in cyber security, especially in the modern era. Its importance and relevance can be clearly seen in the numerous high profile security breaches that have been occurring these days. In this paper we will be studying problems relating to distinguishability. This topic is of great interest and importance to cryptography and other areas. The form of distinguishability that we will be focusing on is static inclusion and its sub-case static equivalence. Our main results are providing saturation based procedures for deciding static equivalence over classes of intruder theories.

I. INTRODUCTION AND MOTIVATION

Distinguishability, whether it is possible to distinguish between two objects A and B, is an important concept in many fields. Some common examples where distinguishability is of crucial importance are:

- the Turing test in AI: can we distinguish between a human and a computer?
- clinical trials: can the patient distinguish between receiving a placebo or the real drug? Can a medical researcher differentiate between the effect of the placebo and the real drug effect?
- journalism: are the sources really multiple (for corroborating a story)? (See also [1].)
- zero knowledge proofs or protocols: can a third party observer determine if (s)he is witnessing a real or a fabricated run of the protocol?
- differential privacy in data analytics: Consider that you have two otherwise identical databases, one which contains your information and one that does not. If we use a statistical query can we determine which database was queried?

A common application of distinguishability in cryptographic protocol analysis is in the detection of guessing attacks. An attack where an intruder makes a guess and is able to verify if the guess is correct is known as a guessing attack [2]. Guessing attacks are also known as dictionary or brute force attacks. A guessing attack is offline when an attacker does not have to interact with other agents to verify his guess [3]. Some examples of protocols that were shown to be susceptible to offline guessing attacks are the Kerberos Authentication protocol [4], Peyravian-Jeffries’s remote user authentication protocol [5], and the Encrypted Key Exchange (EKE) protocol [3].

The main topics that we will explore here are the properties of static inclusion and static equivalence (which is a special case of static inclusion). Static equivalence is an interesting topic in cryptography since it has a significant relation to offline guessing attacks [6, 7, 8]. People often choose weak passwords so the importance of our work is to provide a formal model which captures the knowledge that an intruder gains by observing a protocol. Static equivalence has also been used in analyzing the linkability of e-passports, i.e., whether it is possible (for outsiders/hackers) to check whether the same person has used more than one service such as passport control [9].

Informally static equivalence can be illustrated using the following example which is based on the Dolev-Yao model [10]:

Example I.1. Let Alice and Bob be agents that are trying to communicate in a hostile environment. Consider the following message exchange sequence:

\[
\begin{align*}
\text{Alice} &\rightarrow \text{Bob}: \{M, N_1\}_k \\
\text{Bob} &\rightarrow \text{Alice}: \{M, N_2\}_k
\end{align*}
\]

where \(M\) is a message, \(N_1\) and \(N_2\) are nonces (e.g., randomly generated strings) and \(k\) represents our potentially weak key. Consider that an intruder Eve who has witnessed this message exchange tries to guess a value, \(k'\), for the key and decrypt the above message exchanges using \(k'\). If Eve is then able to determine that the values for \(M\) are the same, then it is highly likely that her guess for \(k\) was correct.

In symbolic protocol analysis, this example is analyzed in terms of substitutions, which are mappings from variables to terms. The messages witnessed by the intruder can be represented by the following substitution:

\[
\theta = \{X_1 \rightarrow e(p(M, N_1), k), \ X_2 \rightarrow e(p(M, N_2), k)\}
\]

where \(e\) represents the encryption operator and \(p\) represents pairing. In this example the set of hidden data contains \(M, N_1, k', N_2, k\). Thus these values (which are treated as symbolic constants) are not available to the intruder.

Note that we will assume that cryptographic primitives are known to all agents including intruders. The intruder's
capabilities can be represented by a set of rewrite rules $R$ as follows:

$$R = \{d(e(x,y), y) \rightarrow x, \pi_1(p(x,y)) \rightarrow x, \pi_2(p(x,y)) \rightarrow y\}$$

where the first rule represents the standard decryption operator, $p$ is a pairing function, and $\pi_i$ is the projection function on the $i^{th}$ element where $i$ is either 1 or 2.

Now suppose that the intruder wants to extend his knowledge by guessing a key $k'$. This can be viewed as an extension of $\theta$:

$$\rho = \theta \cup \{X_1 \rightarrow k\}$$

On the other hand, the principals know the correct key $k$, so their knowledge can be viewed as the substitution

$$\sigma = \theta \cup \{X_1 \rightarrow k\}$$

Clearly, the terms $\pi_1(d(X_1, X_3))$ and $\pi_2(d(X_2, X_4))$ are equivalent under $\sigma$ (and $R$) but not under $\rho$. (In other words, $\sigma$ unifies the terms $\pi_1(d(X_1, X_3))$ and $\pi_2(d(X_2, X_4))$ modulo the rewrite system $R$.) Thus the two frames are distinguishable and they are not statically equivalent.

The terms the intruder can construct are called “recipes” in the literature. In the above example, the terms $\pi_1(d(X_1, X_3))$ and $\pi_2(d(X_2, X_4))$ are both recipes since the intruder can construct them with what (s)he knows. Static inclusion, on the other hand, is the (asymmetric) question of whether $\rho$ can unify every pair that $\sigma$ can. Clearly static equivalence corresponds to the case where static inclusion holds both ways.

Here is an example where there is no guessing attack:

**Example 1.2.** Consider the following protocol:

Alice $\rightarrow$ Bob: \{Na\}_q

Bob $\rightarrow$ Alice: \{Mb, Na\}_K

where $M$ is a message, $Na$ is a nonce, and $q$ represents our potentially weak key and $K$ represents a strong key. [Note that this example is only for illustrative purposes and may not occur in practice since if Alice and Bob already had a strong key, they would not need to use a weak key $q$]

Consider that an intruder Eve who has witnessed this message exchange tries to guess a value, $q'$, for the weak key. As before, the intruder’s capabilities are represented by the above term rewriting system $R$.

The set of hidden data in this example consists of $M$, $Na$, $q$, $q'$, and $K$. The substitutions we have time to apply are

$$\sigma = \{X_1 \rightarrow e(Na, q), X_2 \rightarrow e(p(M, Na), K), X_3 \rightarrow q\}$$

and

$$\rho = \{X_1 \rightarrow e(Na, q), X_2 \rightarrow e(p(M, Na), K), X_3 \rightarrow q\}$$

In this example static equivalence holds and there does not exist a guessing attack. Intuitively we can see that even if the intruder guesses the right value of $q$, (s)he cannot confirm it because the key $K$ used to in the second message is a strong one. We will return to this example later.

Both static inclusion and static equivalence are very important problems in practice; however, static inclusion is somewhat rare in the literature. An application of static inclusion was studied in [11] where it was referred to as static refinement. Static inclusion allows one to compare two protocol implementations with respect to security [11]. Assume for instance Alice sends to Bob the following message: \{Na\}_q, $Na$ where $Na$ is a nonce generated by Alice and $q$ is a key shared by Alice and Bob. A "lazy" implementation of Bob’s role may accept messages like $Na_1, Na_2$. On the other hand a prudent implementation will check the equality of nonces inside and outside the encryption. We note that $\rho = \{X_1 \rightarrow p(e(N, q), Na), X_2 \rightarrow q\}$ is not statically included in $\sigma = \{X_1 \rightarrow p(e(N, q), Na), X_2 \rightarrow q\}$. In [11] a role implementation is considered to be prudent if every message sequence accepted by this role is statically included in the message sequence from its specification (in Alice-Bob notations).

Static inclusion is an undecidable problem in general, even for convergent term rewriting systems [12]. Our purpose in studying static inclusion is to identify decidable sub-cases. Our approach can be seen as an extension to the approach given in “Intruders with Caps” [13]. In that paper the authors consider the deduction problem which is also known as the cap problem. Essentially what this problem asks is if we stack “cap” terms on top of the terms we are considering, can we gain access to the encrypted message? If we can get this message then we call the system treacherous or unsafe. Our approach is very closely based upon this paper. The deduction problem is also undecidable for arbitrary convergent term rewriting systems [12].

An informal outline of our co-saturation procedure:

1) We will have a term rewriting system $R$ that will be used to model some Intruder theories and two substitutions $\sigma$ and $\rho$ (note we are using substitutions for simplicity later we will introduce frames which are substitutions plus a way of keeping track of private data associated with each substitution.)

2) For terms in $\sigma$ and in the range of $\sigma$ we will try to apply one of our two inference rules.

3) If an inference rule is successfully applied a mapping and a mapping has been added to $\sigma$ then in our second substitution $\rho$ we must “mimick” the effect of this inference rule by adding a mapping to $\rho$. Note that these operations are performed in “lockstep.”

4) The addition of these mappings represents the knowledge learned by the honest principals and intruders over the course of a protocol’s runtime.

5) We continue this co-saturation until neither inference rule applies and we terminate or we have hit a failure condition.

6) If we terminate without failure then we perform some additional checks to see if static inclusion holds (i.e., $\sigma$ is statically included in $\rho$).

In the interest of space, proofs have been omitted in this version. For full technical details please see [14].
II. Notation and Preliminaries

We assume the reader is familiar with the usual notions and concepts in term rewriting systems [15]. We also recommend that the reader be familiar with [13]. Throughout this paper we shall refer to a term rewriting system \( R \) (i.e., intruder theory) that is convergent and inter-reduced.

By a "public" symbol we mean a symbol that the intruder has access to: in other words, that is a function that the intruder can apply. All of our function symbols of arity greater than 0 will be public. An \( n \)-ary public symbol \( f \) is said to be transparent for \( R \), or \( R \)-transparent, if and only if its arguments can be extracted by the intruder. Formally, using the notation in [13] \( f \) is \( R \)-transparent if and only if there exist \( n \) \( R \)-cap-terms \( C^n_1(\cdot),\ldots,C^n_n(\cdot) \) such that \( C^n_i(f(s_1,\ldots,s_n)) \rightarrow s_i \), for every \( 1 \leq i \leq n \) where \( s_1,\ldots,s_n \) are distinct variables. A public function symbol is \( R \)-resistant if and only if it is not \( R \)-transparent.

Let \( \succ \) a simplification ordering containing \( R \), i.e., \( l \succ r \) for every rule \( l \rightarrow r \) in \( R \). A rewrite rule \( l \rightarrow r \) is said to be \( \Delta \)-strong, with respect to the simplification ordering \( \succ \), if and only if every \( R \)-resistant subterm of \( l \) is greater than \( r \) with respect to \( \succ \). The intruder term rewriting system \( R \) is said to be \( \Delta \)-strong with respect to \( \succ \) if and only if every rule is \( \Delta \)-strong with respect to \( \succ \). (The ordering \( \succ \) is dropped if it is obvious from the context.)

We define our term algebra as follows: \( T_{\varphi} := T(F \cup \text{Names} \setminus \mathfrak{h}, \text{Dom}(\varphi)) \), where \( \varphi \) is a substitution and \( \mathfrak{h} \) is the set of private constants (i.e., the set of nonces). Intruder knowledge is captured by terms in \( T_{\varphi} \). We shall refer to the terms in \( T_{\varphi} \) as \( \varphi \)-recipes. A term \( t \) that is a \( \varphi \)-instance of a term in \( T_{\varphi} \) is known as a \( \varphi \)-plat 1.

III. Frames, Recipes, and Redundancy

To define static equivalence we shall first need to define the notion of a frame. When we define a frame \( \varphi \) we intuitively mean that we have some private data which we will denote as \( \mathfrak{h} \) and a substitution \( \sigma \) (e.g., a set of mappings). So we define \( \varphi := \forall \mathfrak{h}. \sigma \) where \( \forall \mathfrak{h} \) can be considered as an as a binding (e.g., it is like taking the ordered pair of \((\mathfrak{h},\sigma)\)).

We will now define the notion of a frame more formally. Throughout this section we assume that \( \varphi := \forall \mathfrak{h}. \sigma \) is a frame and that \( V \) is a denumerable set of variable disjoint from \( \text{Dom}(\sigma) \). Furthermore, for ease of exposition, we separate public names, i.e., those in \((\text{Names} \setminus \mathfrak{h})\), into two sets: names that already appear in \( \text{Dom}(\sigma) \) which we call \( \text{Name}_{\text{new}} \) and names that are "brand new". \( n_{\text{new}} \). Thus \( \text{Names} := \mathfrak{h} \uplus \text{Name}_{\text{frame}} \uplus n_{\text{new}} \).

Definition III.1. A term \( t \in T(F \cup \text{Names} \setminus \mathfrak{h}, V \cup \text{Dom}(\sigma)) \) is said to be a general recipe.

In other words, a term \( t \) is a general recipe if and only if one of the following holds:

1) \( t \in V \).
2) \( t \in (\text{Names} \setminus \mathfrak{h}) \).
3) \( t \in \text{Dom}(\sigma) \).
4) \( t = f(t_1,\ldots,t_n) \) where every \( t_i \), \( 1 \leq i \leq n \), is a general recipe.

Definition III.2. A term \( t \in T(F \cup \text{Names}, V) \) is said to be a general plat if and only if one of the following holds:

1) \( t \in V \).
2) \( t \in (\text{Names} \setminus \mathfrak{h}) \).
3) \( t \in \text{Dom}(\sigma) \).
4) \( t = f(t_1,\ldots,t_n) \) where every \( t_i \), \( 1 \leq i \leq n \), is a general plat.

Lemma III.3. A term \( t \in T(F \cup \text{Names}, V) \) is a general plat if and only if there is a general recipe \( s \) such that \( t = s \cdot \sigma(s) \).

Definition III.4. A substitution \( \sigma \) is a plat-extension of \( \varphi \) if and only if

1) \( \sigma \subseteq \sigma \).
2) \( \forall \mathfrak{h} \in (\text{Dom}(\sigma) \setminus \text{Dom}(\varphi)): \sigma(h) \) is a \( \varphi \)-plat.

Lemma III.5. ("Cover Sets") For any term \( t \in T(F \cup \text{Names}, V) \) that has an instance which is a \( \varphi \)-plat, there exists a finite set \( S \) of general recipes such that

1) all terms \( s \in S: \sigma(s) \) is an instance of \( t \), and
2) for all substitutions \( \theta: \forall \mathfrak{h}.(t) \rightarrow T(F \cup \text{Names}) \), if \( \theta(t) \) is a \( \varphi \)-plat, then there exists a general recipe \( T \in S \) and a plat-extension \( \sigma \) of \( \varphi \) such that \( \theta(t) = \sigma(T) \).

This set is referred to as a cover-set for \( t \) with respect to \( \varphi \).

Example III.6. Let \( \forall \mathfrak{h}, \mathfrak{a}, \mathfrak{b}. \{X_1 \rightarrow g_1(a), X_2 \rightarrow f(g_1(a), b)\} \) be a frame and \( t = f(g_1(a), v) \) be a term. The cover-set is \( \{f(g_1(a), v), f(X_1, v), X_2\} \). Note that \( \{f(g_1(a), v)\} \) is not a cover-set by itself, since there is no plat-extension that maps \( f(g_1(a), v) \) to \( f(g_1(a), b) \).

We also consider cover-sets of equations — this essentially amounts to considering equality (=) as a free binary function symbol. Additionally we can consider the cover-set of a convergent term rewriting system \( R \). As \( R \) is a set of rewrite rules \( l \rightarrow r \), we can consider each rule as an equation of the form \( l \equiv_r r \). Now to compute the cover-set of \( R \) we simply take the union of the cover-sets of all such equations. It is worth noting here that for such rules the equation \( l \equiv_r r \) itself will be in the cover-set, since it contains no hidden nonces.

Example III.7. Let \( \forall \mathfrak{a}, \mathfrak{b}. \{X_1 \rightarrow g(a), X_2 \rightarrow h(b)\} \) be a frame and \( f(g(x), h(y)) = g(x) \) be an equation. Its cover-set is

\[
\{f(g(x), h(y)) = g(x), f(X_1, h(y)) = X_1, f(g(x), X_2) = g(x), f(X_1, X_2) = X_1\}.
\]

Now we will provide a definition of \( \mathcal{F} \)-closure. We will use the notions of \( \mathcal{F} \)-closure, \( \mathcal{F} \)-independence and \( \mathcal{F} \)-cores.

1Note that we chose the name \( \text{plat} \) since in French 'plat' means meal. Thus, intuitively, we are building our \( \varphi \)-plats or "meals" from \( \varphi \)-recipes.
when we are performing checks after running our procedures. Intuitively they are used to identify non-redundant information.

**Definition III.8.** Let $S$ be a finite set such that $S \subseteq \tilde{n}$. We define the \textit{$\mathcal{F}$-closure} of $S$, denoted by $\mathcal{F}(S)$, as follows:

- $S \subseteq \mathcal{F}(S)$
- If $f(s_1, \ldots, s_p) \in \mathcal{F}(S)$, then $f(s_1, \ldots, s_p) \in \mathcal{F}(S)$
- Nothing else is in $\mathcal{F}(S)$.

In other words, a term $t \in \tilde{n}$ is in $\mathcal{F}(S)$ if and only if either $t$ itself is in $S$, or the root symbol of $t \in F$ and all its top-level subterms are in $\mathcal{F}(S)$.

**Definition III.9.** A set of terms $\Gamma = \{t_1, \ldots, t_n\}$ is \textit{$\mathcal{F}$-independent} if and only if for all $t_i$, we have $t_i \notin \mathcal{F}(\Gamma \setminus \{t_i\})$. A ground substitution $\theta$ is \textit{$\mathcal{F}$-independent} if and only if $\text{Ran}(\theta)$ is an $\mathcal{F}$-independent set and $\forall t_i, t_j \in \mathcal{R}(\theta) : \theta(t_i) = \theta(t_j) \Leftrightarrow t_i = t_j$. A ground substitution $\theta$ is $\mathcal{F}$-dependent if and only if it is not $\mathcal{F}$-independent.

**Definition III.10.** A set $S_1$ of terms is an \textit{$\mathcal{F}$-core} of a set of terms $S_2$ if and only if $S_1 \subseteq S_2, S_1$ is $\mathcal{F}$-independent and every term in $S_2 \setminus S_1$ belongs to $\mathcal{F}(S_1)$.

Now that we have the notion of $\mathcal{F}$-independence we can define the set of all non-redundant mappings of a substitution $\theta$, which we will denote as $\mathcal{F}$-core$(\theta)$.

**Definition III.11.** A substitution $\theta_i$ is an $\mathcal{F}$-core of a substitution $\theta_j$ if and only if $\theta_i \subseteq \theta_j$. $\theta_i$ is $\mathcal{F}$-independent and $\text{Ran}(\theta_i)$ is an $\mathcal{F}$-core of $\text{Ran}(\theta_j)$.

**Definition III.12.** Let $R$ be a convergent term rewriting system and let $\theta \vdash \psi$. $\sigma$ be a ground substitution and $\psi = \nu \circ \rho$. $\rho$ be frames. A mapping $(x_j \rightarrow s_j)$ in $\sigma$ is syntactically redundant with respect to $\rho$ if and only if there exists a replace $r \in T(F \cup \text{Ran}(\sigma))$ such that $s_j = \sigma(r)$ and $\rho(s_j) = \rho(r)$. $\sigma$ is $\mathcal{F}$-independent and the root symbol of every term in $\mathcal{R}(\sigma)$ is $\mathcal{F}$-independent.

IV. STATIC INCLUSION

Now that we have more notation and preliminaries we will provide a formal definition of static inclusion. We assume that we have the following frames:

Given frames $\phi = \nu \circ \sigma$ and $\psi = \nu \circ \rho$ and an equational theory $\approx$, we say that $\phi$ is statically included in $\psi$ under $\approx$, and write $\phi \subseteq \psi$, if $\forall \theta_0 = \theta_1 \approx \theta_2$ (i.e., $\tilde{\theta} = \tilde{\theta}'$ and $\mathcal{R}(\sigma) = \mathcal{R}(\rho)$) and $\forall t, l \in \text{Ran}(\theta)$, if $\sigma(t) \approx \sigma(l)$ then $\rho(t) \approx \rho(l)$.

Our saturation procedure will require us to extend frames to model the intruder’s growing knowledge over the course of a protocol’s run. A frame $\phi' = \nu \circ \sigma'$ is called a simple extension of $\phi$ if and only if $\sigma \subseteq \sigma'$ and $|\sigma'| = |\sigma| + 1$. In other words, $\sigma$ and $\sigma'$ are of the form

$$\sigma = \{x_1 \rightarrow t_1, x_2 \rightarrow t_2, \ldots, x_n \rightarrow t_n\}$$

$$\sigma' = \sigma \uplus \{x_{n+1} \rightarrow t_{n+1}\}.$$
are the wrong color and even when Harpo switches places with him "inside the mirror." He is nearly convinced that Harpo is his reflection until Chico wanders into the scene and bumps into both Harpo and Groucho. This scene relates to our procedure in many ways. Consider that Groucho is φ and Harpo is ψ. In our procedure if ψ is a clever "mimicker" then φ may still be deceived into thinking that it is statically included in ψ even after our procedure terminates. Thus we must perform additional checks to ensure that φ has not been fooled by clever "mimicking."

We will formulate our co-saturation procedure in Figure 1. Please note that even if the co-saturation exits without failure we cannot deduce that φ is statically included in ψ yet.

**Definition V.4.** We say that two frames φ = vφ,σ and ψ = vψ,ρ are co-saturated if and only if σ does not grow under any application of our inference rules and we have encountered no failures.

**Lemma V.5.** Let R be a convergent term rewriting system and let φ = vφ,σ and ψ = vψ,ρ be co-saturated frames. If (X → g(t_1, ..., t_n)) ∈ σ and ρ is transparent, then (X → g(t_1, ..., t_n)) does not belong to the F-core of σ.

**Lemma V.6.** Let R be a convergent term rewriting system and let φ = vφ,σ and ψ = vψ,ρ be co-saturated frames. Then φ ⊆S ψ if and only if every mapping in σ not in its F-core is syntactically redundant with respect to ρ.

**Lemma V.7.** Any frames φ = vφ,σ and ψ = vψ,ρ can be co-saturated in a finite number of steps.

**Example V.8.** Let R = {d(x,y), y → x}. Let φ = vφ,σ = v[k, k_1], X → e(a, k), X_2 → k_1 and ψ = vψ,ρ = v[k, k_1], X → e(a, k), X_2 → k_1 be frames. Saturation yields the same substitutions as before:

\[ \sigma = \{ X_1 → e(a, k), X_2 → k, X_3 → a \} \]

\[ \rho = \{ X_1 → e(a, k), X_2 → k, X_3 → d(e(a, k), k_1) \} \]

The F-core of σ is \( X_2 → k \) since a is a public constant. The mapping \( X_1 → e(a, k) \) is not in F-core of σ and is not syntactically redundant in ρ, since \( X_1 = σ(e(a, X_2)) \) and \( X_3 ≠ ρ(e(a, X_2)) \). Thus by Lemma V.6 φ ⊈S ψ. (The mapping X_3 → a is not redundant either.)

**Example V.9.** (A small variation of Example V.8)

Let \( \mathcal{R} = \{ d(x,y), y → x, d(x,y), y → x \} \) and let \( \phi = vφ,σ = v[k, k_1], \{ X_1 → e(a, k), X_2 → k \} \) and \( \psi = vψ,ρ = v[k, k_1], \{ X_1 → e(a, k), X_2 → k_1 \} \) be frames. Saturation yields the same substitutions as before:

\[ \sigma = \{ X_1 → e(a, k), X_2 → k, X_3 → a \} \]

\[ \rho = \{ X_1 → e(a, k), X_2 → k, X_3 → d(e(a, k), k_1) \} \]

The F-core of σ is \( X_2 → k \) since a is a public constant. The mapping \( X_1 → e(a, k) \) now is syntactically redundant in ρ, since \( X_1 = σ(e(X_3, X_2)) \) and
\[ p(e(X_1, X_2)) = e(d(e(a, k), k_1), k_2) \rightarrow^* e(a, k) = p(X_1). \]

**Example V.10.** Let \( R = \{ f(g(x), y) \rightarrow h(x) \}. \) Let \( \theta = \forall \alpha. \sigma = \forall \alpha. \{ X_1 \rightarrow g(a) \} \) and \( \psi = \forall \alpha. \rho = \forall \alpha. \{ X_1 \rightarrow h(a) \} \) be frames. The \( s \)-overlap is \( \theta(l) = f(g(a), y). \) Note that \( \theta(l) = \sigma(f(X_1, y)) \), thus \( \theta(l) \) is a \( \Phi \)-flat and \( \theta(r) = h(a) \notin \text{An} (\sigma). \) The cover-set of \( \theta(l) \) is

\[ \mathcal{F}_\theta(\theta(l)) = \{ f(X_1, y) \} \]

\[ \mathcal{U} = \rho(\{ f(X_1, y) \}) \rightarrow^* \mathcal{G}(f(h(a), y)). \] Since \( \theta(l) \) is not a ground singleton set, \( \Phi \not\subseteq \psi. \)

**Example V.11.** Let \( R = \{ f(g(x), y), g(x, z) \rightarrow h(x) \}. \) Let

\[ \theta = \forall \alpha. \sigma = \forall \alpha. \{ X_1 \rightarrow g(a, b), X_2 \rightarrow g(a, c) \} \]

and

\[ \psi = \forall \alpha. \rho = \forall \alpha. \{ a, b, c, d, d', a \} \cdot X_1 \rightarrow g(d, b), X_2 \rightarrow g(d', c) \]

be frames.

Consider the \( s \)-overlap is \( \theta(l) = f(g(a, b), g(a, z)). \) Note that 
\[ \{ z \rightarrow b \} \theta(l) = \sigma(f(X_1, X_1)), \] thus \( \theta(l) \) has an instance that is a \( \Phi \)-flat. \( \theta(r) = \theta(h(x)) = h(a) \notin \text{An} (\sigma). \) The cover-set of \( \theta(l) \) is

\[ \mathcal{F}_\theta(\theta(l)) = \{ f(X_1, X_1), f(X_1, X_2) \} \]

\[ \mathcal{U} = \rho(\{ f(X_1, X_1), f(X_1, X_2) \}) \rightarrow^* \mathcal{G}(h(d), f(g(d, b), g(d', c))). \]

Since \( \mathcal{U} \) is not a ground singleton set, we conclude \( \Phi \not\subseteq \psi. \)

Please note that in [14] we have provided this co-saturation procedure in more detail along a more general version of this co-saturation procedure which can handle a larger class of intruder theories. We only provided the above co-saturation procedure since it is a bit easier to understand and due to space constraints.

**VI. RELATED WORK**

Static inclusion and its sub-case static equivalence have been studied from many directions. Research has been going on in these and related problems for some time. Part of the reason why these types of problems have been studied so much is because humans tend to choose very poor and often easily guessable passwords [18].

Static equivalence decision procedures have been proposed for various equational theories [12, 19, 20] including sub-term convergent theories. Yet Another Protocol Analysis Tool (YAPA) [21], Knowledge in Security protocols (KiSs) [22], and FAST [23] can verify static equivalence for a large set of equational theories; however the precise set of theories under which these algorithms terminate is not clear. ProVerif [24] is a general cryptographic protocol analysis tool that can verify equivalence properties even in presence of active attackers, but without termination guarantees.

Trace equivalence, a more general form of static equivalence, has been used to show that many protocols are in fact, insecure. One such protocol is Basic Access Control (BAC) that was used for French passport authentication [25, 26]. An interesting, and perhaps surprising, note is that this protocol became insecure due to the addition of certain security checks. When these checks were added the protocol’s response time was not padded (i.e., successful and unsuccessful runs could have different run times). Thus if an adversary was able to time the response of various runs anonymity could be compromised.

It can be noted that in static inclusion we assume that we have a passive intruder. A passive intruder is assumed to be able to eavesdrop on message exchanges between agents and use common cryptographic primitives. However, they are not allowed to interact with the principals. A more practical model of an intruder is an active intruder. A problem that is related to static inclusion that allows for an active intruder is known as observational equivalence [27]. Observational equivalence is stronger than both static equivalence and trace equivalence. Due to this, it is often more difficult to check. Thus, more focus has gone into researching static equivalence and trace equivalence.

**VII. CONCLUSION AND FUTURE WORK**

The main contribution that we have made is providing a co-saturation procedure for deciding if a frame \( \phi \) is statically included in another frame \( \phi \) over specific classes of intruder theories. We developed our approach by essentially starting where “Intruders with Caps” ended. In that paper the authors were studying a related problem, the cap or deduction problem. We modified their procedures to solve the decision problem for static inclusion over \( \Delta \)-strong theories. For this class of theories we guarantee termination, soundness, and completeness.

A notion we introduced that was not in [13] is that of a cover-set. This was not needed in [13] because exactly how a term was deduced was not important there. Deciding the deduction problem was shown to be non-primitive recursive for general \( \Delta \)-strong intruder theories [28]. The same proof can be adapted to show that deciding static inclusion is also of non-primitive recursive complexity over general \( \Delta \)-strong intruder theories (see [14]).

Another goal is to generalize our current technique to handle more classes of intruder theories. Additionally we would like to implement our co-saturation procedure and include it in a protocol analysis tool such as Maude-NPA [29]. We plan to provide detailed comparisons to other implementations such as YAPA [21], KiSs [22], and FAST [23].
REFERENCES


