Soundness of Removing Cancellation Identities in Protocol Analysis under Exclusive-OR

Sreekanth Malladi

Dakota State University
Madison, SD 57042, USA
Sreekanth.Malladi@dsu.edu

Abstract. In [Mil03, LM05], Millen-Lynch-Meadows proved that, under some restrictions on messages, including identities for canceling an encryption and a decryption within the same term during analysis will be redundant. i.e., they will not lead to any new attacks that were not found without them. In this paper, we prove that slightly modified restrictions are sufficient to safely remove those identities, even when protocols contain operators such as the notorious Exclusive-OR operator that break the free algebra assumption with their own identities, in addition to the identities considered by Millen-Lynch-Meadows.

Keywords: Cryptographic protocol analysis, Free algebras, Equational theories, Constraint solving, Exclusive-OR.

1 Introduction

1.1 Background

Consider the following protocol:

$$\begin{array}{lll} \text{Message 1.} & A \rightarrow B & : [K_{AB}]_{sh(A,B)} \\ \text{Message 2.} & C \rightarrow A & : [N]_{sh(B,C)} \\ \text{Message 3.} & A \rightarrow B & : [[N]_{sh(B,C)}]_{sh(A,B)} \\ \text{Message 4.} & B \rightarrow C & : N. \end{array}$$

(*Notation*: A, B, C are agent variables; K_{AB} is a session-key variable; N is a nonce variable; $sh(X, Y)$ represents the long-term shared-key of agents X and Y; $[t]_k$ represents t encrypted with k using a symmetric cipher).

Suppose A and B are played by two honest agents a and b, while C is played by a dishonest agent c. Then the following attack is possible:

$$\begin{array}{lll} \text{Message 1.} & a \rightarrow b & : [k_{ab}]_{sh(a,b)} \\ \text{Message 2.} & c \rightarrow a & : [n]_{sh(b,c)} \\ \text{Message 3.} & c(a) \rightarrow b & : [k_{ab}]_{sh(a,b)} \quad \text{(replaying Message 1)} \\ \text{Message 4.} & b \rightarrow c & : [k_{ab}]^{-1}_{sh(b,c)}. \end{array}$$

(*Note*: We use all lower-case symbols now since this is a protocol execution. $c(a)$ denotes c spoofing as a).

S. Mödersheim and C. Palamidessi (Eds.): TOSCA 2011, LNCS 6993, pp. 205–224, 2012.

The protocol and the attack were inspired from the example in [Mil03]. Basically, in the attack, c replays the first message to b as the third message. Since b does not know that the plain-text of the encryption is a session-key k_{ab}, not the encryption that it was expecting, it would innocently decrypt it using $sh(a, b)$, then with $sh(b, c)$, and send $[k_{ab}]^{-1}_{sh(b,c)}$ to[1] c, thinking it sent n. Agent c can happily encrypt it with $sh(b, c)$ that it shares with b, to learn k_{ab}.

This vulnerability could be found when $[K_{AB}]_{sh(A,B)}$ and $[[N]_{sh(B,C)}]_{sh(A,B)}$ are unified, whence it would be apparent that N should be substituted with $[K_{AB}]^{-1}_{sh(B,C)}$. However, we would not discover it, if the unification algorithm did not include the identity $[[p]^{-1}_r]_r = p$. i.e., an Explicit Decryption Operator (EDO) identity.

Protocol analysis techniques that adopt an identity-free (or free algebra) model such as [THG98, MS01, HS02] miss attacks that exploit identities. Others like the NRL analyzer [Mea92, Mea96] that do include such identities would discover them.

Millen considers this issue in [Mil03] and notes that, if protocols are designed without terms like $[\_]^{-1}_\_$ (called *pure* protocols) and[2] do not contain terms of the form $[X]_\_$ at all where X is a variable (called *EV-Free* protocols), then we would never have terms of the form $[[\_]^{-1}_\_]_\_$ during analysis, in which case, the EDO identity is never used. For instance, in the above protocol, if message 3 is changed to $[[N, B]_{sh(B,C)}]_{sh(A,B)}$, agent B would expect to see a pair after decrypting with $sh(A, B)$ and $sh(B, C)$. If not, it would reject the message. Hence, the attack exploiting the EDO identity would not exist.

Similarly, Lynch & Meadows extended Millen's result to the asymmetric encryption case [LM05]. They showed that if protocols do not use private encryption keys or public signature keys explicitly in messages and do not contain terms of the form[3] $[X]^{\rightarrow}_\_$, then cancellations in asymmetric encryptions will not be possible and the corresponding identities, e.g. $[[t]^{\rightarrow}_{pv(a)}]^{\rightarrow}_{pk(a)} = t$ will be redundant during analysis. They call such protocols, *pk-pure*, and *PEV-free*.

The point here is not that the example attack above is a realistic scenario, but the conditions in which an identity-free analysis is sound. Indeed, it is well-known that encryptions should have some redundancy to ensure correct decryption and to use some random number in the plain-text as well, since deterministic encryption is insecure. Millen-Lynch-Meadows show that the same principles are sufficient for identity-free analysis, and hence their restrictions are not additional — they are a must anyway for secure protocols. But the lesser

[1] $[t]^{-1}_k$ denotes t decrypted with k using a symmetric cipher; the notation might seem strange, but it is motivated by the fact that decryption in symmetric ciphers uses the same key as encryption, but the process is inverted (or reversed).

[2] Following Lowe [Low99], we use underscore (_) when the value in that place is irrelevant in a formula; this helps the reader to focus on the important values.

[3] We use a superscript $\rightarrow$ to indicate the use of an asymmetric cipher, because the key inverse is used for encryption/decryption, but the process is the same as encryption (there is not a reverse process for decryption).

the identities used in protocol analysis, the easier and faster analysis becomes, so Millen-Lynch-Meadows results are quite useful for protocol analysis.

1.2 The Problems and Our Contributions

We wanted to find out whether Millen-Lynch-Meadows results hold when more operators are used in protocols that have their own identities, in addition to those considered by Millen-Meadows-Lynch. For instance, the Exclusive-OR (XOR) operator is one such operator that possesses ACUN (Associativity, Commutativity, existence of Unity and Nilpotence) identities. To be precise, the main questions we were concerned about were:

1. *Can we still conduct protocol analysis without the identities considered by Millen-Meadows-Lynch when protocols use operators such as XOR that have their own identities?*
2. *Are the restrictions given by Millen-Meadows-Lynch sufficient in such protocols or do we need more restrictions?*

It is very important to find these out since many protocols used in real-life such as SSH and SSL use the XOR operator. This operator is also notorious since its ACUN identities were used to reveal surprising attacks that were not discovered without them [RS98]. Further, contemporary research efforts are focusing on these kind of problems. For instance, in a very recent work [SEMM10], the theory of ACUN with public-key/private-key cancellation was used to show new attacks. [SEMM10] also emphasized the importance of studying combinations of theories, quite strongly.

We found that EV-freedom doesn't necessarily help under XOR. For instance, if the third message in our example protocol was $[[N \oplus B]_{sh(B,C)}]_{sh(A,B)}$, it is EV-Free. But the attack is still possible, by replaying message 1 into message 3, since b cannot check the format of the XOR term inside. In fact, b would decrypt the replayed message with $sh(a, b)$, then with $sh(b, c)$, and send $[k_{ab}]^{-1}_{sh(b,c)} \oplus b$ as the fourth message (thinking it sent just n). Agent c can obtain k_{ab} by xoring it with b, followed by encrypting with $sh(b, c)$.

But if the third message in the protocol is changed to some $[[1, N \oplus B]_{sh(B,C)}]_{sh(A,B)}$, the attack is thwarted, since B would look for a pair after decrypting with $sh(A, B)$ and $sh(B, C)$. Using this concept, we make the following contributions:

1. A combination and unification of the results in [Mil03] and [LM05] who dealt with symmetric and asymmetric encryption respectively, but not both;
2. We show that when protocols adopt a new, slightly modified version of EV-Freedom, called **EVX-Freedom**, and the other restriction of Millen-Lynch-Meadows called *purity*, using identities to cancel encryptions do not reveal any new attacks under XOR (Section 3);
3. We also show that if protocols obey another scheme that is a slight modification of a scheme in [LM05] called **Structure** (independent of purity and EVX-Freedom), cancellation identities again do not reveal any new attacks;
4. We fix a few minor errors in [Mil03] and [LM05].

2 Protocol Model

In this section, we will define the term algebra in Section 2.1, attacker deductions in Section 2.2, and constraint solving for protocol analysis in Section 2.3.

2.1 Term Algebra

We assume the existence of a set of variables denoted *Vars*. The signature Σ contains the set of nullary functions symbols denoted *Constants* and another set of symbols to construct more terms: $\{tuple, senc, sdec, penc, sign, xor\}$.

senc and *sdec* represent symmetric encryption and decryption operators respectively. *penc* represents asymmetric key encryption. There is no *pdec*, since as we noted before, in most asymmetric ciphers, the process of decryption is the same as encryption but using the inverse key. *sign* represents the signature of a term such that $sign(t, k)$ is a signature of t that is to be verified using the inverse of k.

We will call the terms created with operators *senc*, *sdec*, *penc* and *sign* as "encrypted terms". We will also use a predicate encrypted() that returns true only if the argument supplied to it is an encrypted term.

Cumulatively, we define the infinite set *Terms* as,

$$Terms = Vars \cup Constants \cup \{f(t_1, \ldots, t_n) \mid f \in \Sigma \wedge t_1, \ldots, t_n \in Terms\}.$$

Syntactically, $tuple(t_1, \ldots, t_n) = [t_1, \ldots, t_n]$, $senc(t, k) = [t]_k$, $sdec(t, k) = [t]_k^{-1}$, $penc(t, k) = [t]_k^{\rightarrow}$, $sign(t, k) = [t]_k^{\mapsto}$, $xor(t_1, t_2) = t_1 \oplus t_2$.

We use a superscript $\mapsto$ for signatures, since although they use asymmetric ciphers, they are different from asymmetric encryptions, since they encrypt a hash of the text, not the text itself.

In contrast to [Mil03, LM05] who used a functional notation throughout, we use the above syntactic sugar to denote terms, which we believe helps in following the proofs. For instance, [LM05] denotes the asymmetric encryption of X with a private-key K_1 and public-key K_2 as $pe(pk(K_2, pub, enc), pe(pk(K_1, priv, enc), X))$, which might be easier to follow if denoted as $[[X]_{pv(K_1)}^{\rightarrow}]_{pk(K_2)}^{\rightarrow}$.

We will call t in $[t]_k$ or $[t]_k^{\rightarrow}$ or $[t]_k^{\mapsto}$ as the "plain-text" of those terms and k as their "key". We use functions *plaintext*() and *key*() to refer to the plain-text and key of encrypted terms.

Using AC properties of *xor*, we write $xor(t_1, xor(t_2, t_3))$ simply as $t_1 \oplus t_2 \oplus t_3$.

The *subterm* relation $\sqsubseteq$ is defined as,

$$(t' \sqsubseteq t) \text{ iff } (t' = t) \vee (\exists f \in \Sigma; t_1, \ldots, t_n; t'') \left(\begin{array}{c} (t = f(t_1, \ldots, t_n)) \wedge \\ (t'' \in \{t_1, \ldots, t_n\}) \wedge (t' \sqsubseteq t'') \end{array} \right).$$

We will denote the subterms of a term t as $SubTerms(t)$. We will denote the subterms of t that are encrypted as $EncSubTerms(t)$. A *ground term* is a term with no variables as subterms.

By $pk(k)$ and $spk(k')$ we denote the keys k and k' being used as public-keys for asymmetric encryption. Similarly, $pv(k)$ and $spv(k')$ as private keys. The use of functions pk, pv, spk, spv helps in distinguishing the purpose of the keys and define the identities and deductions accordingly. For instance, our identities and deductions will not allow $[t]^{\rightarrow}_{pk(k)}$ to be decrypted with $spv(k)$.

Note also that $pk(k)$ (and similarly $pv(k)$, $spk(k)$, $spv(k)$) does not mean that k is necessarily an agent identity; it can be any term. i.e., $pk(k)$ is not necessarily an atomic key where $pk()$ is a look-up function on agents' public-keys. It only denotes k being used is a key for asymmetric encryption that is known to everyone and that it possesses an inverse denoted $pv(k)$ that is known to some or only one agent.

We define the following set of identities that reflect cancellation of asymmetric/symmetric encryption/decryption rounds, denoted E_1=$\{E_{S1}, E_{S2}, E_{P1}, E_{P2}, E_{P3}, E_{P4}\}$ where,

$$E_{S1} : [[t]_k^{-1}]_k = t,\ E_{P1} : [[t]^{\rightarrow}_{pk(k)}]^{\rightarrow}_{pv(k)} = t,\ E_{P3} : [[t]^{\mapsto}_{spv(k)}]^{\mapsto}_{spk(k)} = t$$
$$E_{S2} : [[t]_k]_k^{-1} = t,\ E_{P2} : [[t]^{\rightarrow}_{pv(k)}]^{\rightarrow}_{pk(k)} = t,\ E_{P4} : [[t]^{\mapsto}_{spk(k)}]^{\mapsto}_{spv(k)} = t$$

We also give the ACUN identities, denoted $E_2 = \{E_A, E_C, E_U, E_N\}$ where,

$$E_A : t_1 \oplus (t_2 \oplus t_3) = (t_1 \oplus t_2) \oplus t_3,\ E_U : t \oplus 0 = t,$$
$$E_C : t_1 \oplus t_2 = t_2 \oplus t_1, \qquad E_N : t \oplus t = 0.$$

We denote $E = E_1 \cup E_2$. We will denote by $R/R_1/R_2$ the rewriting rules to be applied on a term to reduce it using the identities $E/E_1/E_2$ respectively. R can be shown to be confluent. R_1 can be shown to be convergent using techniques described in [Mea92]. R_2 will not be convergent because of E_C. Hence, when we refer to irreducibility under E_2, we mean the irreducibility under E_2 modulo E_A, E_C. We will denote the normal form of a term t (or set of terms) modulo a set of rewriting rules R as $t \downarrow_R$.

The main results in this paper show that R_1 are inapplicable on terms under some syntactic restrictions. This would mean that when R are applied on terms, effectively only R_2 are applied.

We will use a predicate `irred`(,) taking parameters a term/set of terms/substitution and a set of identities/rewrite rules that returns `true` if the former is irreducible modulo the latter. E.g. If t is a term, `irred`(t, R_1) is `true` if t is irreducible modulo R_1.

2.2 Attacker Deductions

Our attacker deduction model is based on Lowe's model in [Low04]. We model single step attacker deduction rules through the relation $\vdash$, defined as:

$$\vdash\ :: \mathcal{P}(Terms) \times Labels \times Terms$$

such that, $S \vdash_l s$ represents that the attacker can deduce s from S using the action label l belonging to the set $Labels$.

We use two different sets of deduction rules to achieve two different results in Sections 3 and 4. We define them separately in those sections.

We define the *derivation* of a term s from a set of terms S using deduction rules L, identities E that are represented by rewrite rules R, using the relation $\models$:

Let S be irreducible by R. Then,

$$S \models_{E,L} s \Leftrightarrow (\exists \langle S_1 \vdash s_1, \ldots, S_n \vdash s_n \rangle) \left((\forall i \in \{1, \ldots, n\}) \left(\begin{array}{l} (S_1 = S) \wedge (s_n = s) \wedge \\ (S_{i+1} \subseteq S_i \cup \{s_i \downarrow_R\}) \wedge \\ (\exists l \in L; \sigma; T \vdash_l t)((T\sigma =_E S_i\sigma \wedge t\sigma =_E s_i\sigma)). \end{array} \right) \right). \quad (1)$$

Read $S \models_{E,L} s$ as s is *derivable* from S using the deduction rules L and identities E.

2.3 Protocol Analysis Using Constraint Solving

We now define strands to model protocols and constraint solving for protocol analysis.

Definition 1 (Node, Strand, Protocol, Semi-bundle)
A node *is a tuple* $\langle \pm, t \rangle$*, where* $+$ *and* $-$ *denote "sending" and "receiving" a term* t *respectively. A* strand *is a sequence of nodes. A* protocol *is a set of strands. A* semi-bundle *is a collection of strands from a protocol, after applying some substitutions to some of the variables in the strands.*

We will overload the function *SubTerms*() to return all the subterms in a set of strands.

A constraint is denoted $m : T$ where m is a term and T is a set of terms. Protocol analysis on semi-bundles using constraint solving can reveal vulnerabilities on protocols.

Definition 2 (Constraints, Satisfiability)
A constraint $m : T$ *is* satisfiable *using a substitution* σ*, identities* E *and deduction rules* $\mathcal{D}$ *if* $T\sigma$*,* $m\sigma$ *are ground terms, and* $T\sigma \models_{E,\mathcal{D}} m\sigma$*:*

$$\mathsf{satisfiable}(\sigma, E, \mathcal{D}, m : T) \Leftrightarrow T\sigma \models_{E,\mathcal{D}} m\sigma.$$

A constraint sequence $C = \langle m_1 : T_1, \ldots, m_n : T_n \rangle$ *is from a semi-bundle* S *if*

- *every* $m : T \in C$ *is such that*
  - m *is a term on a receiving node in* S*;*
  - *every* $t \in T$ *is a term on a sending node in* S*;*
  - *if* m *and* t *are on the same strand, then* t *precedes*[4] m*;*
- *for all* m_i*,* m_j *where* $i, j \in \{1, \ldots, n\}$*,* m_i *precedes* m_j *if they are both on the same strand;*
- $(\forall i \in \{1, \ldots, n-1\})(T_i \subseteq T_{i+1})$.

[4] t_i *precedes* t_j if $\langle t_1, \ldots, t_n \rangle$ is a strand and $t_i, t_j \in \{t_1, \ldots, t_n\}$ s.t. $i < j$.

C is satisfiable *with σ under $(E, \mathcal{D})$, iff every constraint in C is satisfiable with the same substitution:*

$$\mathsf{satisfiable}(\sigma, E, D, C) \Leftrightarrow (\forall c \in C)(\mathsf{satisfiable}(\sigma, E, \mathcal{D}, c)).$$

We denote all possible constraint sequences from a semi-bundle S as $ConSeq(S)$.

We assume that every protocol has a set of variables that are intended to be kept secret in each execution of the protocol. We denote them $SecVars(Pr)$. We also denote the constants substituted to those variables as $secrets(S)$ if S is a semi-bundle of Pr such that $S = Pr\sigma_S$ for some substitution σ_S.

A protocol has an attack on secrecy if a constraint sequence from a semi-bundle of a protocol after an artificial constraint with a secret as the target term to the end of the constraint sequence is satisfiable:

Definition 3 (Insecurity for secrecy)
A protocol Pr is insecure for secrecy *under identities E and deduction rules $\mathcal{D}$ if*

- *$C = \langle \_ : \_, \ldots, \_ : T \rangle \in ConSeq(S)$ such that S is a semi-bundle of Pr and*
- *$C^\frown\langle m : T \rangle$ is[5] satisfiable where $m \in secrets(S)$.*

i.e.,

$$\mathsf{insecureForSecrecy}(Pr, E, \mathcal{D}) \Rightarrow \left(\begin{array}{c} (\exists \sigma; \sigma_S; C; S) \\ (S = Pr\sigma_S) \wedge (m \in secrets(S)) \wedge \\ (C = \langle \_ : \_, \ldots, \_ : T \rangle \in ConSeq(S)) \wedge \\ \mathsf{satisfiable}(\sigma, E, \mathcal{D}, C^\frown\langle m : T \rangle) \end{array} \right).$$

3 Purity and EVX-Freedom Imply Soundness

In this section, we will prove that syntactic restrictions called purity and EVX-Freedom on terms are sufficient to ensure that no new attacks can be found by exploiting identities E_1 given in Section 2.1.

3.1 Attacker Deduction Rules, $\mathcal{D}$ and $\mathcal{DE}$

We first give rules $\mathcal{D}$ and $\mathcal{DE}$ that we will consider, starting with $\mathcal{D}$:

$$\begin{array}{lll} [t_1, \ldots, t_n] \vdash_{ex_i} t_i & \{t, k\} \vdash_{senc} [t]_k & \{t, pk(k)\} \vdash_{pkenc} [t]^{\rightarrow}_{pk(k)} \\ \{t_1, \ldots, t_n\} \vdash_{comb} [t_1, \ldots, t_n] & \{[t]_k, k\} \vdash_{sdec} t & \{[t]^{\rightarrow}_{pk(k)}, pv(k)\} \vdash_{pkdec} t \\ \{t_1, t_2\} \vdash_{xor} t_1 \oplus t_2 & \{t, spv(k)\} \vdash_{pvsigenc} [t]^{\mapsto}_{spv(k)} & \{[t]^{\mapsto}_{spv(k)}, spk(k)\} \vdash_{pvsigdec} t \end{array}$$

Rules in $\mathcal{DE}$ include all the ones in $\mathcal{D}$ and some additional ones below:

$$\{t, k\} \vdash_{sdenc} [t]^{-1}_k \quad \{t, pv(k)\} \vdash_{pvenc} [t]^{\rightarrow}_{pv(k)} \quad \{t, spk(k)\} \vdash_{pksigenc} [t]^{\mapsto}_{spk(k)}$$

[5] $^\frown$ is the sequence concatenation operator.

Notice that we don't have the following deductions:

- $\{[t]_k^{-1}, k\} \vdash_{sddec} t$, since it can be simulated with *senc* and the identity $E_{S1} : [[t]_k^{-1}]_k = t$.
- $\{[t]^{\rightarrow}_{pv(k)}, pk(k)\} \vdash_{pvdec} t$ since it can be simulated with *penc*, and E_{P2}; similarly, *pksigdec*, which can be simulated with *pvsigenc* and E_{P4} respectively;
- $\{t_1 \oplus t_2, t_1\} \vdash t_1$ and others involving $\oplus$, that can be simulated with *xor* and E_2.

3.2 Purity and EVX-Freedom Requirements

We now define the syntactic requirements of purity and EVX-Freedom on protocols that we claim void the deductions $\mathcal{DE} \setminus \mathcal{D}$ and identities E_1 during protocol analysis.

Definition 4 (Purity and EVX-Freedom).
A term t is `pure` *if it does not have subterms of the form* $[\_]_\_^{-1}$, $pv(\_)$ *or* $spk(\_)$*:*

$$\texttt{pure}(t) \Leftrightarrow (\nexists [\_]_\_^{-1}, pv(\_), spk(\_) \sqsubseteq t).$$

A term t is `EVX-Free` *if the plain-texts of all its encrypted subterms are not variables or* `XOR` *terms:*

$$\texttt{EVXF}(t) \Leftrightarrow (\forall t' \sqsubseteq t)(plaintext(t') \notin Vars \wedge plaintext(t') \neq \_ \oplus \ldots \oplus \_).$$

The definition of purity might seem "non-uniform": since it prohibits private encryption keys $pv(\_)$, it might seem natural to prohibit private signature keys $spv(\_)$ as well. But we need to permit either $spv(\_)$ or $spk(\_)$, not both, in order to prevent the use of identities E_{P3} and E_{P4}. We choose to keep $spv(\_)$, since one uses $spk(k)$ to verify $[\_]^{\mapsto}_{spv(\_)}$, not vice-versa. If we prohibit $spv(\_)$, we would have to expect people to possess other's private signature keys!

Another alternative is to simply allow only terms of the form $[t]^{\mapsto}_{spk(k)}$ in the term algebra, with the intention that it represents that the signature of t is verified with $spk(k)$. But in that case, we would have to remove any deductions in $\mathcal{D}$ and $\mathcal{DE}$ that use $spv(\_)$, which is not unreasonable since signatures are only verified — decrypting them will not serve any purpose since people encrypt the hash of messages when signing, not the message itself.

We will call a protocol Pr as `pure` or `EVX-Free` if $SubTerms(Pr)$ are `pure` or `EVX-Free` respectively. We will assume that if a protocol is pure or EVX-Free, then every semi-bundle from it is also pure and EVX-Free.

3.3 Soundness Proofs

In this section we will prove our main claim. We will start with a few lemmas that assist us in the proof of the main theorem at the end.

The first lemma is the most crucial, lynchpin lemma.

Lemma 1 (Purity and EVX-Freedom ensure irreducibility under R_1) *Let t be a term and σ a substitution. Then,*

$$\mathtt{pure}(t) \wedge \mathtt{EVXF}(t) \wedge \mathtt{irred}(\sigma, R) \Rightarrow (t\sigma) \downarrow_R = (t\sigma) \downarrow_{R_2} .$$

Proof. A term is reducible under R_1 if it has a subterm that resembles:

- $[[\_]^{-1}_{\_}]_{\_}$, $[[\_]^{\rightarrow}_{pv(\_)}]^{\rightarrow}_{pk(\_)}$, $[[\_]^{\rightarrow}_{pk(\_)}]^{\rightarrow}_{pv(\_)}$, $[[\_]^{\mapsto}_{spv(\_)}]^{\mapsto}_{spk(\_)}$, $[[\_]^{\mapsto}_{spk(\_)}]^{\mapsto}_{spv(\_)}$; or
- $[t]_{\_}$, $[t]^{\rightarrow}_{pk(\_)}$, $[t]^{\mapsto}_{spv(\_)}$, such that a substitution to the term t can result in the term matching a term in the first case, after rules in R_2 are applied.

But both these cases are not possible when a term is pure and EVX-Free by definition.

Note that purity and EVX-Freedom do not prevent reducibility under R_2, but only R_1. For instance, as in the example given in the Introduction, $[1, X \oplus A \oplus b]_k$ is pure and EVX-Free, but it is reducible under R_2 with a substitution $\sigma = \{b/A\}$. But this is not of concern to us, since we are not trying to prove that our syntactic restrictions prevent attacks that exploit ACUN identities, but only the ones that exploit cancellation of symmetric/asymmetric encryption/decryption operations (R_1).

Next, we define a new rewrite system P that "purifies" a term by replacing all subterms of the form $[X]^{-1}_Y$ to $[X]_Y$. It also replaces those of the form $[X]^{\rightarrow}_{pv(Y)}$, $[X]^{\mapsto}_{spk(Y)}$ with X, since the decryption keys for those terms are publically available anyway.

$$\begin{array}{ll} \textbf{(a)} & [t]^{-1}_k \rightarrow [t]_k \\ \textbf{(b)} & [t]^{\rightarrow}_{pv(k)} \rightarrow t \\ \textbf{(c)} & [t]^{\mapsto}_{spk(k)} \rightarrow t \end{array}$$

P can be applied on any impure term to convert it into a pure term. Obviously, every pure term t is irreducible modulo P. i.e., if t is pure, then $t \downarrow_P = t$. Also, every pure term is irreducible with R_1.

The idea behind defining purification is to subsequently use it to show that any breach of security in an "impure" analysis (using $\mathcal{DE}, E$) can also be simulated in a "pure" analysis (using $\mathcal{D}, E_2$) after purifying the terms.

The next lemma is similar to Lemma 1 except that only purity of a term is assumed, not EVX-Freedom.

Lemma 2 (Irreducibility preserved under purification) *Let t be a term and σ be a substitution. Then,*

$$\mathtt{pure}(t) \wedge \mathtt{irred}(\sigma, P) \Rightarrow \mathtt{irred}(t\sigma, P).$$

Proof. Under the symmetric case, since t is pure, it does not have a subterm t' such that t' resembles $[\_]^{-1}_{\_}$. Since σ is irreducible, it does not have such a term either. Hence, $t\sigma$ will not have such terms and hence is irreducible under P.

Similarly, for the asymmetric case, being pure, t does not have any private keys or public signature keys as subterms. Since σ doesn't have such terms either (being irreducible under P), we have that $t\sigma$ is irreducible under P.

Corollary 1. *We can infer from the lemma that if t and σ are irreducible under P, it does not matter whether we purify $t\sigma$ with P or first purify σ with P and then apply it to t. i.e., $(t\sigma)\downarrow_P = t(\sigma\downarrow_P)$.*

The next step is to show that every single deduction in $\mathcal{DE}$ on a set of terms S that are irreducible under R_1 is also possible in $\mathcal{D}$ when necessary (i.e., if the deduced term is not already in S), if S has been purified using P.

Theorem 1 (Deductions preserved under purification)
Let S be a set of terms. Then,

$$\begin{pmatrix} (S \vdash_l s) \wedge (l \in \mathcal{DE}) \wedge \\ \texttt{irred}(S, R_1) \end{pmatrix} \Rightarrow \begin{pmatrix} (\exists l' \in \mathcal{D})(S\downarrow_P \vdash_{l'} s\downarrow_P) \vee \\ (s\downarrow_P \in S\downarrow_P). \end{pmatrix}.$$

Proof (Sketch)
The detailed proof can be found in Appendix A, Theorem 4.

We show that for each deduction in $\mathcal{DE}$, there is an equivalent deduction in $\mathcal{D}$.

Deductions that deduce terms irreducible under R_1 are trivial. Some of these are, *ex_i*, *comb*, *xor*, and *pkdec*.

For the other deductions, there are two possibilities: either the deduced term is irreducible under R_1 or reducible:

- If it is irreducible, we have a corresponding deduction in $\mathcal{D}$ when S and s are purified. For instance, consider the following deduction using the rule *sdenc* in $\mathcal{DE}$: $\{t, k\} \vdash_{sdenc} [t]_k^{-1}$.
  If $\texttt{irred}([t]_k^{-1}, R_1)$, then we can have a deduction using rule *senc* in $\mathcal{D}$:

$$\{t\downarrow_P, k\downarrow_P\} \vdash_{senc} [t\downarrow_P]_{k\downarrow_P}.$$

- If it is reducible, then we have another deduction in $\mathcal{D}$ which simulates the combination of the equation that it is reducible with, and the deduction in $\mathcal{DE}$. For instance, in the same example as the first case, suppose $t = [t']_k$ (here t' must be irreducible under R_1, since t is). Then, $[[t']_k]_k^{-1} =_{E_1} t'$. Hence, for the deduction below:

$$\{[t']_k, k\} \vdash_{sdenc} [[t']_k]_k^{-1}$$

  we have the following deduction using rule *sdec* in $\mathcal{D}$:

$$\{[t']_{k\downarrow_P}, k\downarrow_P\} \vdash_{sdec} t'.$$

We are now ready to achieve the main result for this section.

Theorem 2 (Main Result 1)
If a protocol Pr is pure and EVX-Free and if Pr is insecure for secrecy under $(E, \mathcal{DE})$, it is also insecure for secrecy under $(E_2, \mathcal{D})$.

Proof (Sketch)
The detailed proof is in Appendix A. Briefly, since Pr is insecure for secrecy, from Def. 3, we have that a constraint sequence C of a semi-bundle from Pr is satisfiable after an artificial constraint with its target as a secret term is added to C.

From Theorem 1, we have that if a constraint in C is satisfied using $\models_{E,\mathcal{DE}}$, it can also be satisfied using $\models_{E_2,\mathcal{D}}$, if all the terms in C are purified using the purifying rewrite system P.

Hence, we have that Pr is also insecure for secrecy under $(E_2, \mathcal{D})$.

4 Structured Protocols

We now define the other requirement on protocols, namely "structured terms":

Definition 5. *A term is* structured *iff the plaintext of each of its encrypted subterms is not a variable, an XOR term or an encrypted term;*

$$\mathsf{structured}(t) \Leftrightarrow (\forall t' \sqsubseteq \mathit{EncSubTerms}(t)) \begin{pmatrix} (\mathit{plaintext}(t') \notin \mathit{Vars}) \wedge \\ (\mathit{plaintext}(t') \neq \_ \oplus \ldots \oplus \_) \wedge \\ \neg\mathsf{encrypted}(\mathit{plaintext}(t')) \end{pmatrix}.$$

A protocol Pr is structured or $\mathsf{structured}(Pr)$ *iff every term in $\mathit{SubTerms}(t)$ is structured.*

A simple way to ensure that terms are structured is by adding constants as tags to plain-texts of all encryptions that are not tuples, along the lines of [HLS03, RS05, ML09].

Structured protocols need not be pure and they are EVX-Free by definition. Structure achieves the effects of both purity and EVX-Freedom in preventing cancellations.

Lack of purity allows cancellation after substitution. For instance, $[[t]^{-1}_{X\oplus b}]_k$ which will not occur in pure protocols is reducible when $X = k \oplus b$. Similarly, for asymmetric encryptions, for pure protocols, $[[\_]^{\rightarrow}_{pv(\_)}]^{\rightarrow}_{pk(\_)}$ do not occur, and $[[\_]^{\rightarrow}_{spv(\_)}]^{\rightarrow}_{pk(\_)}$ that could occur do not cancel because we assume $spv(k) \neq pv(k)$, for all k and $spk(k) \neq pk(k)$ as well. In this section, structure inhibits such cancellations even in impure terms. We show that attacks found on structured protocols using R can also be found using R_2 alone, with additional deductions to make up for R_1.

4.1 Attacker Deductions

Like in Section 3, we first define the attacker deduction rules that we will consider. The first set of rules is $\mathcal{DS}$, which are used only with E_2. These include those in $\mathcal{D}$ and some additional ones below:

$$\{t, k\} \vdash_{sdenc} [t]^{-1}_k \quad \{t, pv(k)\} \vdash_{pvenc} [t]^{\rightarrow}_{pv(k)} \quad \{t, spk(k) \vdash_{pksigenc} [t]^{\mapsto}_{spk(k)}$$

These were not part of $\mathcal{D}$ because of purity, but they were a part of $\mathcal{DE}$, because they were needed to be used in conjunction with R_1.

$\mathcal{DS}$ also includes two more rules below:

$$\{[t]^{\rightarrow}_{pv(k)}, pk(k)\} \vdash_{pvdec} t \quad \{[t]^{\mapsto}_{spk(k)}, spv(k)\} \vdash_{pksigdec} t$$

These were not in $\mathcal{D}$ because such terms do not appear in pure terms. But now we do not assume purity. They were also not included in $\mathcal{DE}$ because they can be simulated by combining a deduction rule and a reduction rule in R_1.

We define $\mathcal{DSE}$ to include those in $\mathcal{DS}$ but excluding *pkdec*, *pvsigdec*, *pvdec*, and *pksigdec*. $\mathcal{DSE}$ are used along with the identities E for deducing terms. The excluded rules can be simulated with other rules and E. For instance, *pkdec* - $\{[t]^{\rightarrow}_{pk(k)}, pv(k)\} \vdash_{pkdec} t$ can be simulated with *pvenc* - $\{t, pv(k)\} \vdash_{pvenc} [t]^{\rightarrow}_{pv(k)}$ by using $[t]^{\rightarrow}_{pk(k)}$ in place of t, whence, $[t]^{\rightarrow}_{pv(k)}$ becomes $[[t]^{\rightarrow}_{pk(k)}]^{\rightarrow}_{pv(k)} = t$ by virtue of E_{P1}.

4.2 Proofs

We first prove a lemma that is analogous to Lemma 1.

Lemma 3 (Irreducibility by R_1 preserved for structured protocols) *Let t be a term and σ a substitution. Then,*

$$\mathsf{structured}(t) \wedge \mathtt{irred}(\sigma, R) \Rightarrow (t\sigma) \downarrow_R = (t\sigma) \downarrow_{R_2} .$$

Proof. The only way $t\sigma$ can be reducible under R_1 is if it was of the form $[[\_]^{-1}_{\_}]_{\_}$, $[[\_]^{\rightarrow}_{pv(\_)}]^{\rightarrow}_{pk(\_)}$, $[[\_]^{\rightarrow}_{pk(\_)}]^{\rightarrow}_{pv(\_)}$, $[[\_]^{\mapsto}_{spv(\_)}]^{\mapsto}_{spk(\_)}$, $[[\_]^{\mapsto}_{spk(\_)}]^{\mapsto}_{spv(\_)}$.

But these forms are possible only if t had an encrypted subterm of these forms or an encrypted subterm with its plaintext a variable, `XOR` term or an encrypted term so that $t\sigma$ will resemble those forms. None of these are possible when t is a structured term.

Hence, we have, $(t\sigma) \downarrow_R = (t\sigma) \downarrow_{R_2}$.

We are now ready to prove the main theorem on structured protocols. We do not need to use any purification of terms using P now, since the theorem applies for pure and impure protocols alike.

There was an error in the proof of a similar theorem in [LM05, Theorem 3]. They claim they consider the deduction *pkenc*, but they actually consider *pvenc* (most likely inadvertently). We fix it here.

Theorem 3 (Main Result 2)
If a structured protocol Pr is insecure for secrecy under $(E, \mathcal{DSE})$, it is insecure for secrecy under $(E_2, \mathcal{DS})$.

Proof. We first prove that if S is a set of terms irreducible under R_1, then if a term s can be deduced from S using rules in $\mathcal{DSE}$, it can also be deduced using rules in $\mathcal{DS}$:

$$(\forall l \in \mathcal{DSE}) \left(\mathtt{irred}(S, R_1) \wedge (S \vdash_l s \downarrow_{R_1}) \Rightarrow (\exists l' \in \mathcal{DS})(S \vdash_{l'} s) \right). \qquad (2)$$

This is straightforward for deductions like ex_i, *comb*, and *xor*, which deduce terms that are irreducible under R_1. For instance, consider *comb*, which is possible in $\mathcal{DSE}$ and $\mathcal{DS}$ alike, since

$$\{s_1 \downarrow_{R_1}, s_2 \downarrow_{R_1}\} \vdash_{comb} (s_1 \downarrow_{R_1}, s_2 \downarrow_{R_1})$$

is the same as $\{s_1, s_2\} \vdash_{comb} (s_1, s_2)$, given that $\mathtt{irred}(S, R_1)$ from hypothesis.

For the other deductions, we have to show that any result of a deduction in $\mathcal{DSE}$ that is reducible under R_1 can be deduced using a deduction rule in $\mathcal{DS}$ that was removed in $\mathcal{DSE}$.

For instance, consider *senc*:

$$\{t \downarrow_{R_1}, k \downarrow_{R_1}\} \vdash_{senc} [t]_k \downarrow_{R_1},$$

where $[t]_k$ is reducible by R_1. Then, t must be some $[t']_k^{-1}$ where t' and k are irreducible. So the deduction is now: $\{[t']_k^{-1} \downarrow_{R_1}, k \downarrow_{R_1}\} \vdash_{senc} [[t']_k^{-1}]_k \downarrow_{R_1} = t'$.

But this is the same as: $\{[t']_k^{-1}, k\} \vdash_{sddec} t'$.

(recall that *sddec* belongs to $\mathcal{DS}$, but not $\mathcal{DSE}$).

Similarly, consider *penc*: If $[t]^{\rightarrow}_{pk(k)}$ is reducible to t' such that $t = [t']^{\rightarrow}_{pv(k)}$, then

$$\{t \downarrow_{R_1}, pk(k \downarrow_{R_1})\} \vdash_{penc} [t]^{\rightarrow}_{pk(k)} \downarrow_{R_1}$$

is the same as $\{[t']^{\rightarrow}_{pv(k)}, pk(k)\} \vdash_{pvdec} t'$.

Similar reasoning applies for *pvsigenc* which can be simulated by *pksigdec* if the result $[t]^{\rightarrow}_{spv(k)}$ were to be reducible.

We have included a formal version of the rest of the proof in Appendix A, since it is similar to Theorem 2.

Informally, since Pr is insecure for secrecy, from Def. 3, we have that a constraint sequence C of a semi-bundle from Pr is satisfiable after an artificial constraint with its target as a secret term is added to C.

From above, we have that if a constraint in C is satisfied using $\models_{E,\mathcal{DSE}}$, it can also be satisfied using $\models_{E_2,\mathcal{DS}}$, if all the terms in C are purified using the purifying rewrite system P.

Hence, we have that Pr is also insecure for secrecy under $(E_2, \mathcal{DS})$.

5 Related Work

There has been a great amount of interesting research published in the last decade combining algebraic properties with intruder deductions. For instance, Basin et al. present a uniform and modular approach to handling algebraic properties in protocol analysis [BMV05]. A good coverage of such results is given in the survey by Cortier et al. [CDL06].

It seems that research in protocol analysis under algebraic properties at this point is split largely into two directions:

- Results showing how the properties can be included while developing analysis or verification tools. Extensions to the constraint solver, Maude-NPA, ProVerif and other tools are being undertaken world-wide in this direction;
- Parallel results showing how to safely remove some identities, aiding in the development and effective use of tools.

Our contributions in this paper clearly fall in the second category. While it is not possible to cover all the related articles here, below are some results in the same spirit:

- In a very significant result, Comon and Delaune describe how the presence of the "finite variant property" which holds for theories such as Abelian groups ensures that some algebraic properties can be safely removed [CD05];
- In [KT08], Kuesters-Truderung demonstrate the $\oplus$-linearity property for protocols to reduce the protocol verification problem to free algebra verification, when verifying using ProVerif;
- In [ML09, Mal10] we have shown that under the restriction of tagging messages (similar to the tagging in this paper), the role of ACUN identities during unification is restricted so that unifiers result only from syntactic unification. The net effect being that removing type-flaw and multi-protocol attacks from consideration during analysis is sound.

6 Conclusion

In this paper, we proved that every attack found by including cancellation identities for symmetric and asymmetric encryption/decryption can also be found without them, under some reasonable syntactic restrictions on protocols, even when they use operators like Exclusive-OR that possess their own identities.

The basic concept behind EV-Freedom [Mil03], PEV-Freedom [LM05], EVX-Freedom, and Structure is the same: Protocols should be designed so that agents will be able to verify some property of messages after decryption, such as their number in the protocol, operators used to create them etc. This is a prudent engineering practice [AN94], has been used to guarantee protocol security against important forms of attacks [HLS03, GT00, ML09, Mal10] and ensure decidability [Low99, RS05].

Exclusive-OR is just one of many similar operators and theories that we can extend the results under. Especially, theories that are disjoint with the standard algebra[6] like monoidal theories can be similarly considered and the main results can be easily achieved. Only Lemmas 1 and 3 would have to be changed for other theories.

Identities such as $[a, b] = a$, $[a, b] = a \oplus b$ etc. cannot be similarly considered, since the lynchpin Lemmas 1 and 3 cannot be extended when E_2 contains those equations. But such identities are usually quite unrealistic and impractical, so

[6] i.e., those that do not use free operators like tuples in their identities.

it is probably not worth the trouble to invent restrictions that would preserve soundness of analysis under them anyway.

As pointed out in [LM05], our results on purity and EVX-Freedom cannot be extended directly to other trace properties, in particular, authentication. This is because, they involve purification that alters the protocol specification. For instance, authentication of an agent that she is indeed Alice might depend on determining if the agent possesses $[secret]^{\rightarrow}_{pv(\mathsf{Alice})}$ according to the protocol. But our main theorem 1 in Section 3 purifies the term into *secret*, removing the encryption layer. Therefore, its derivation in a purified protocol doesn't imply that authentication is violated under $(E_2, \mathcal{D})$ in the original protocol as well, since it is sent in plain anyway. This is not a problem for the main theorem 2 on structured protocols though, since there is no purification or alteration of protocol there. That result can be extended to authentication by a suitable definition for that property, analogous to Def. 3 for secrecy.

The results in this paper are part of an ongoing effort to scale the security analysis of protocols hierarchically following [Mea03], starting with the most basic model. Extensions of results would require extending proofs in the base model appropriately with additional operators, theories and attacker capabilities. While Millen's result [Mil03] was the initial base for this result, Lynch-Meadows was the next step in the hierarchy. Our contribution is about extending both [Mil03] and [LM05] appropriately with an additional equational theory, and extending their proofs accordingly by strategically introducing the theory at crucial points.

We believe that future research in protocols will have to be similarly conducted, building on the work in the base model, extending it to the desired level in the hierarchy.

Acknowledgments. I am greatly indebted to Chris Lynch (Clarkson University) for helpful comments on the paper, and many clarifications on [LM05]. Thanks also to the anonymous reviewers whose comments helped in improving the presentation.

This work was conducted as a result of funding from the International Institute of Information Technology (IIIT) during summer of 2010. I am grateful to Kishore Kothapalli, Bezawada Bruhadeshwar (faculty, CSTAR, IIIT) and Prof. Rajeev Sangal (Director, IIIT) for their support.

References

[AN94] Abadi, M., Needham, R.: Prudent Engineering Practice for Cryptographic Protocols. In: Proc. IEEE Symposium on Research in Security and Privacy, pp. 122–136. IEEE Computer Society Press, Los Alamitos (1994)

[BMV05] Basin, D., Mödersheim, S., Viganò, L.: Algebraic intruder deductions. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 549–564. Springer, Heidelberg (2005)

[CD05] Comon-Lundh, H., Delaune, S.: The finite variant property: How to get rid of some algebraic properties. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 294–307. Springer, Heidelberg (2005)

[CDL06] Cortier, V., Delaune, S., Lafourcade, P.: A of algebraic properties used in cryptographic protocols. Journal of Computer Security 14(1), 1–43 (2006)

[GT00] Guttman, J.D., Thayer, F.J.: Protocol Independence through Disjoint Encryption. In: 13th IEEE Computer Security Foundations Workshop, pp. 24–34 (July 2000)

[HLS03] Heather, J., Lowe, G., Schneider, S.: How to prevent type flaw attacks on security protocols. Journal of Computer Security 11(2), 217–244 (2003)

[HS02] Heather, J., Schneider, S.: Equal to the task? In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 162–177. Springer, Heidelberg (2002)

[KT08] Küsters, R., Truderung, T.: Reducing protocol analysis with xor to the xor-free case in the horn theory based approach. In: ACM Conference on Computer and Communications Security, pp. 129–138 (2008)

[LM05] Lynch, C., Meadows, C.: On the relative soundness of the free algebra model for public key encryption. Electr. Notes Theor. Comput. Sci. 125(1), 43–54 (2005)

[Low99] Lowe, G.: Towards a completeness result for model checking of security protocols. Journal of Computer Security 7(2-3), 89–146 (1999)

[Low04] Lowe, G.: Analysing protocols subject to guessing attacks. Journal of Computer Security 12, 83–98 (2004)

[Mal10] Malladi, S.: Protocol independence through disjoint encryption under exclusive-or. In: Workshop on Foundations of Computer Security and Privacy, FCSPrivMod (2010)

[Mea92] Meadows, C.: Applying formal methods to the analysis of a key management protocol. Journal of Computer Security 1(1), 5–36 (1992)

[Mea96] Meadows, C.: Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. In: Bertino, E., Kurth, H., Martella, G., Montolivo, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 351–364. Springer, Heidelberg (1996)

[Mea03] Meadows, C.: Towards a hierarchy of cryptographic protocol specifications. In: FMSE 2003: Formal Methods in Security Engineering. ACM Press, New York (2003)

[Mil03] Millen, J.: On the Freedom of Decryption. Information Processing Letters 86(6), 329–333 (2003)

[ML09] Malladi, S., Lafourcade, P.: How to prevent type-flaw attacks under algebraic properties. In: Workshop on Security and Rewriting Techniques, Affiliated to CSF 2009 (July 2009)

[MS01] Millen, J., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: Proc. ACM Conference on Computer and Communication Security, pp. 166–175. ACM Press, New York (2001)

[RS98] Ryan, P.Y.A., Schneider, S.A.: An attack on a recursive authentication protocol. a cautionary tale. Inf. Process. Lett. 65(1), 7–10 (1998)

[RS05] Ramanujam, R., Suresh, S.P.: Decidability of context-explicit security protocols. Journal of Computer Security 13, 135–165 (2005)

[SEMM10] Sasse, R., Escobar, S., Meadows, C., Meseguer, J.: Protocol analysis modulo combination of theories: A case study in maude-npa. To Appear, Sixth International Workshop on Security and Trust Management (STM). ERCIM (European Research Consortium in Informatics and Mathematics) (2010)

[THG98] Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: Why is a security protocol correct? In: Proc. IEEE Symposium on Research in Security and Privacy, pp. 160–171. IEEE Computer Society Press, Los Alamitos (1998)

A Detailed Proofs

Theorem 4 (Deductions preserved under purification) *Let S be a set of terms. Then,*

$$\begin{pmatrix} (S \vdash_l s) \wedge (l \in \mathcal{DE}) \wedge \\ \texttt{irred}(S, R_1) \end{pmatrix} \Rightarrow \begin{pmatrix} (\exists l' \in \mathcal{D})(S \downarrow_P \vdash_{l'} s \downarrow_P) \vee \\ (s \downarrow_P \in S \downarrow_P). \end{pmatrix}.$$

Proof. We will prove the theorem for each deduction in $\mathcal{DE}$.

ex_i, comb: For deductions ex_i and *comb*, that every deduction in $\mathcal{DE}$ has a corresponding deduction in $\mathcal{D}$ is obvious. For instance, if $\{s_1, s_2\} \vdash_{comb} [s_1, s_2]$, then there can be a corresponding deduction in $\mathcal{D}$ such that $\{s_1 \downarrow_P, s_2 \downarrow_P\} \vdash_{comb} [s_1 \downarrow_P, s_2 \downarrow_P]$.

senc: Suppose $\{t, k\} \vdash_{senc} [t]_k$. Then, we have two cases:

$\texttt{irred}([t]_k, R_1)$: In this case, we can have a deduction $\{t \downarrow_P, k \downarrow_P\} \vdash_{senc} [t \downarrow_P \ ]_{k \downarrow_P}$.

$\neg\texttt{irred}([t]_k, R_1)$: In this case, since $\texttt{irred}(t, R_1)$, suppose $t = [t']_k^{-1}$, where $\texttt{irred}(t', R_1)$. Then, $[t]_k =_{E_1} [[t]_k^{-1}]_k = t'$. We will then use *sdec* on $t \downarrow_P$, $k \downarrow_P$ such that $\{t \downarrow_P, k \downarrow_P\} = \{[t']_k^{-1} \downarrow_P, k \downarrow_P\} = \{[t' \downarrow_P]_{k \downarrow_P}, k \downarrow_P\} \vdash_{sdec} t'$.

sdec: If $\{[t]_k, k\} \vdash_{sdec} t$, then $\texttt{irred}(t, R_1)$ since $\texttt{irred}([t]_k, R_1)$. Therefore, we can have a deduction $\{[t \downarrow_P]_{k \downarrow_P}\} \vdash_{sdec} t \downarrow_P$.

sdenc: If $\{t, k\} \vdash_{sdenc} [t]_k^{-1}$, we have two cases:

$\texttt{irred}([t]_k^{-1}, R_1)$: Then, we can have a deduction $\{t \downarrow_P, k \downarrow_P\} \vdash_{senc} [t \downarrow_P]_{k \downarrow_P}$.

$\neg\texttt{irred}([t]_k^{-1}, R_1)$: Then, suppose $t = [t']_k^{-1}$, where $\texttt{irred}(t', R_1)$, since $\texttt{irred}(t, R_1)$. Then, $[[t']_k]_k^{-1} =_{E_1} t'$. Hence, for $\{[t']_k, k\} \vdash_{sdenc} [[t']_k]_k^{-1}$, we have $\{[t']_{k \downarrow_P}, k \downarrow_P\} \vdash_{sdec} t'$.

xor: If $\{t_1, t_2\} \vdash_{xor} t_1 \oplus t_2$, then we can have: $\{t_1 \downarrow_P, t_2 \downarrow_P\} \vdash_{xor} t_1 \downarrow_P \oplus t_2 \downarrow_P$.

pkenc: $\{t, pk(k)\} \vdash_{pkenc} [t]^{\rightarrow}_{pk(k)}$. We have two cases:

$\texttt{irred}([t]^{\rightarrow}_{pk(k)}, R_1)$: Then, we have $\{t \downarrow_P, pk(k \downarrow_P)\} \vdash_{pkenc} [t \downarrow_P]^{\rightarrow}_{pk(k \downarrow_P)}$.

$\neg\texttt{irred}([t]^{\rightarrow}_{pk(k)}, R_1)$: Let $t = [t']^{\rightarrow}_{pv(k)}$. Then, $\{[t']^{\rightarrow}_{pv(k)} \downarrow_P, pk(k \downarrow_P)\} = \{t' \downarrow_P, pk(k \downarrow_P)\}$ which implies $t' \downarrow_P \in S \downarrow_P$. Similar reasoning applies for *pvsigenc*.

pkdec: Suppose, $\{[t]^{\rightarrow}_{pk(k)}, pv(k)\} \vdash_{pkdec} t$. Then, we have, $\{[t \downarrow_P]^{\rightarrow}_{pk(k \downarrow_P)}, pv(k \downarrow_P)\} \vdash_{pkdec} t \downarrow_P$.
Similar reasoning applies for *pvsigdec*.

pvenc: Suppose, $\{t, pv(k)\} \vdash_{pvenc} [t]^{\rightarrow}_{pv(k)}$. We have two cases:

$\texttt{irred}([t]^{\rightarrow}_{pv(k)}, R_1)$: Then, $[t]^{\rightarrow}_{pv(k)} \downarrow_P = t \downarrow_P$, which belongs to $\{t \downarrow_P, pv(k \downarrow_P)\}$.
$\neg\texttt{irred}([t]^{\rightarrow}_{pv(k)}, R_1)$: Then, let $t = [t']^{\rightarrow}_{pk(k)}$. So we can have, $\{[t' \downarrow_P]^{\rightarrow}_{pk(k\downarrow_P)}, pv(k \downarrow_P)\} \vdash_{pkdec} t' \downarrow_P$. Similar reasoning applies for *pksigenc*.

Theorem 5 (Main Result 1)
If a protocol Pr is pure and EVX-Free and if Pr is insecure for secrecy under $(E, \mathcal{DE})$*, it is also insecure for secrecy under* $(E_2, \mathcal{D})$.

Proof. Since Pr is insecure for secrecy under $(E, \mathcal{DE})$, from Def. 2, suppose $C = C'^{\frown}\langle m \ : \ T_n\rangle$, where $m \in secrets(S)$, $S = Pr\sigma_S$, $C' = \langle m_1 \ : \ T_1, \ldots, m_n \ : \ T_n\rangle \in ConSeq(S)$ where:

$$(\exists\sigma)\left((\forall m \ : \ T \in C)(T\sigma \models_{E,\mathcal{DE}} m\sigma)\right). \tag{3}$$

Consider a constraint $m \ : \ T \in C$. Let $T\sigma = X$, and $m\sigma = x$. If $T\sigma \models_{E,\mathcal{DE}} x$, then by the definition of $\models$ (Eq. 1) we have:

$$\left(\begin{array}{c}(\exists\langle S_1 \vdash s_1, \ldots, S_n \vdash s_n\rangle)\\ (\forall i \in \{1, \ldots, n\})\left(\begin{array}{l}(S_1 = X) \wedge (s_n = x) \wedge\\ (S_{i+1} \subseteq S_i \cup \{s_i \downarrow_R\}) \wedge\\ (\exists l \in \mathcal{DE}; \sigma; T \vdash_l t)(T\sigma =_E S_i\sigma \wedge t\sigma =_E s_i\sigma).\end{array}\right)\end{array}\right). \tag{4}$$

Since m and T are `pure`, `EVX-Free`, if σ is irreducible modulo R, from Lemma 1 we have:

$$(\forall t \in T\sigma \cup \{m\sigma\})(t \downarrow_R = t \downarrow_{R_2}). \tag{5}$$

which implies $\texttt{irred}(T\sigma \cup \{m\sigma\}, R_1)$ and also that

$$(\forall t \in T\sigma \cup \{m\sigma\})((t =_E t') \Rightarrow (t =_{E_2} t')). \tag{6}$$

Combining (4) with (5), (6) and Theorem 1 we have:

$$\left(\begin{array}{c}(\exists\langle S'_1 \vdash s'_1, \ldots, S'_p \vdash s'_p\rangle)\\ (p \leq n) \wedge\\ (\forall i \in \{1, \ldots, p\})\left(\begin{array}{l}(S'_1 = X \downarrow_P) \wedge (s'_p = x \downarrow_P) \wedge\\ (S'_{i+1} \subseteq S'_i \cup \{s_i \downarrow_{R_2}\}) \wedge\\ (\exists l' \in \mathcal{D}; \sigma; T \vdash_{l'} t)(T\sigma =_{E_2} S'_i\sigma \wedge t\sigma =_{E_2} s'_i\sigma).\end{array}\right)\end{array}\right). \tag{7}$$

which implies $X \downarrow_P \models_{E_2,D} x \downarrow_P$. i.e., $(T\sigma) \downarrow_P \models_{E_2,D} (m\sigma) \downarrow_P$.
Using this in Corollary 1 we have, $T\sigma \downarrow_P \models_{E_2,\mathcal{D}} m\sigma \downarrow_P$.
Applying this in (3) above, we have:

$$(\exists\sigma)\left((\forall m \ : \ T \in C)(T\sigma \downarrow_P \models_{E_2,\mathcal{D}} m\sigma \downarrow_P)\right). \tag{8}$$

From Def. 3, this implies that Pr is insecure for secrecy under $(E_2, \mathcal{D})$. Hence, the result.

Theorem 6 (Main Result 2)
If a structured protocol Pr is insecure for secrecy under $(E, \mathcal{DSE})$, it is insecure for secrecy under $(E_2, \mathcal{DS})$.

Proof. We first prove that if S is a set of terms irreducible under R_1, then if a term s can be deduced from S using rules in $\mathcal{DSE}$, it can also be deduced using rules in $\mathcal{DS}$:

$$(\forall l \in \mathcal{DSE})\left(\texttt{irred}(S, R_1) \wedge (S \vdash_l s \downarrow_{R_1}) \Rightarrow (\exists l' \in \mathcal{DS})(S \vdash_{l'} s)\right). \quad (9)$$

This is straightforward for deductions like ex_i, *comb*, and *xor*, which deduce terms that are irreducible under R_1. For instance, consider *comb*, which is possible in $\mathcal{DSE}$ and $\mathcal{DS}$ alike, since

$$\{s_1 \downarrow_{R_1}, s_2 \downarrow_{R_1}\} \vdash_{comb} (s_1 \downarrow_{R_1}, s_2 \downarrow_{R_1})$$

is the same as $\{s_1, s_2\} \vdash_{comb} (s_1, s_2)$, given that $\texttt{irred}(S, R_1)$ from hypothesis.

For the other deductions, we have to show that any result of a deduction in $\mathcal{DSE}$ that is reducible under R_1 can be deduced using a deduction rule in $\mathcal{DS}$ that was removed in $\mathcal{DSE}$.

For instance, consider *senc*:

$$\{t \downarrow_{R_1}, k \downarrow_{R_1}\} \vdash_{senc} [t]_k \downarrow_{R_1},$$

where $[t]_k$ is reducible by R_1. Then, t must be some $[t']_k^{-1}$ where t' and k are irreducible. So the deduction is now:

$$\{[t']_k^{-1} \downarrow_{R_1}, k \downarrow_{R_1}\} \vdash_{senc} [[t']_k^{-1}]_k \downarrow_{R_1} = t'.$$

But this is the same as: $\{[t']_k^{-1}, k\} \vdash_{sddec} t'$ (recall that *sddec* belongs to $\mathcal{DS}$, but not $\mathcal{DSE}$).

Similarly, consider *penc*: If $[t]^{\rightarrow}_{pk(k)}$ is reducible to t' such that $t = [t']^{\rightarrow}_{pv(k)}$, then $\{t \downarrow_{R_1}, pk(k \downarrow_{R_1})\} \vdash_{penc} [t]^{\rightarrow}_{pk(k)} \downarrow_{R_1}$ is the same as $\{[t']^{\rightarrow}_{pv(k)}, pk(k)\} \vdash_{pvdec} t'$.

Similar reasoning applies for *pvsigenc* which can be simulated by *pksigdec* if the result $[t]^{\rightarrow}_{spv(k)}$ were to be reducible.

Since Pr is insecure for secrecy under $(E, \mathcal{DSE})$, from Def. 2, suppose $C = C'^{\frown}\langle m : T_n\rangle$, where $m \in secrets(S)$, $S = Pr\sigma_S$, $C' = \langle m_1 : T_1, \ldots, m_n : T_n\rangle \in ConSeq(S)$ such that:

$$(\exists \sigma)\left((\forall m : T \in C)(T\sigma \models_{E,\mathcal{DSE}} m\sigma)\right). \quad (10)$$

Consider a constraint $m : T \in C$. Let $T\sigma = X$, and $m\sigma = x$. If $T\sigma \models_{E,\mathcal{DSE}} x$, then by the definition of $\models$ (Eq. 1) we have:

$$\left(\begin{array}{c}(\exists \langle S_1 \vdash s_1, \ldots, S_n \vdash s_n\rangle)\\ (\forall i \in \{1, \ldots, n\})\left(\begin{array}{l}(S_1 = X) \wedge (s_n = x) \wedge\\ (S_{i+1} \subseteq S_i \cup \{s_i \downarrow_R\}) \wedge\\ (\exists l \in \mathcal{DSE}; \sigma)((T \vdash_l t \wedge\\ (T \cup \{t\})\sigma =_E (S_i \cup \{s_i\})\sigma)).\end{array}\right)\end{array}\right). \quad (11)$$

Since m and T are structured, if σ is irreducible modulo R, from Lemma 3 we have:

$$(\forall t \in T\sigma \cup \{m\sigma\})(t \downarrow_R = t \downarrow_{R_2}). \tag{12}$$

which implies $\texttt{irred}(T\sigma \cup \{m\sigma\}, R_1)$ and also that

$$(\forall t \in T\sigma \cup \{m\sigma\})((t =_E t') \Rightarrow (t =_{E_2} t')). \tag{13}$$

Combining (11) with (12), (13) and (9) we have:

$$\begin{array}{c}(\exists \langle S'_1 \vdash s'_1, \ldots, S'_n \vdash s'_n \rangle) \\ \left((\forall i \in \{1, \ldots, n\}) \left(\begin{array}{l} (S'_1 = X) \wedge (s'_n = x) \wedge \\ (S'_{i+1} \subseteq S'_i \cup \{s_i \downarrow_{R_2}\}) \wedge \\ (\exists l' \in \mathcal{DS}; \sigma)((T \vdash_{l'} t \wedge \\ (T \cup \{t\})\sigma =_{E_2} (S'_i \cup \{s'_i\})\sigma)). \end{array} \right) \right).\end{array} \tag{14}$$

which implies $X \models_{E_2,DS} x$. i.e., $T\sigma \models_{E_2,DS} m\sigma$.
Applying this in (10) above, we have:

$$(\exists \sigma) \left((\forall m \ : \ T \in C)(T\sigma \models_{E_2,\mathcal{DS}} m\sigma) \right). \tag{15}$$

From Def. 3, this implies that Pr is insecure for secrecy under $(E_2, \mathcal{DS})$. Hence, the result.