On the Freedom of Decryption*

Jonathan Millen
Computer Science Laboratory, SRI International
Menlo Park, CA 94025

December 27, 2002

Abstract

Some formal methods for cryptographic protocol analysis represent message fields using a free term algebra, which does not permit an explicit symmetric decryption operator. Although the ability of principals and intruders to decrypt encrypted messages is represented implicitly, such models can fail to recognize some attacks. However, with an additional restriction on the protocol – EV-freedom, in which encrypted message fields must have a known structure – the extension of the free algebra with decryption is unnecessary because it does not enable any new attacks. The analogous question for public key encryption is open.

1 Introduction

Formal methods for security protocol analysis represent cryptographic operations abstractly, using a logical or algebraic model. Some approaches use a free term algebra to represent messages. A free algebra model has no relations, so a free algebra with an encryption operator cannot also have a decryption operator, since in that case one would have a relation like $\mathrm{d}(\mathrm{e}(X, K), K) = X$. Free term algebras are used in the strand space work [THG99] and approaches based on it, as well as in some other approaches, such as [Pau98]. A good discussion of some of the assumptions and consequences of using a free algebra, such as unique readability, is given in [THG98]. Some analysis tools, such as the NRL analyzer described in [Mea92], are not restricted to a free algebra, and are capable of analyzing protocols that employ decryption explicitly, such as the selective broadcast protocol analyzed in that paper.

Using a free algebra, the ability of legitimate parties and the intruder to decrypt messages is modeled as a transformation from an encrypted expression like $\mathrm{e}(X, K)$, using K, to the decrypted content X. The ability to represent decryption implicitly leads to the plausible supposition that nothing would be

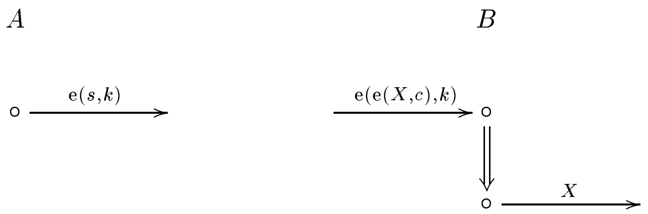
*This work was supported by DARPA under Space and Naval Warfare Systems Center contract no. N66001-00-C-8014.

gained by adding a decryption operator explicitly, if the protocol can be specified without it.

Unfortunately, this is not the case; some attacks are overlooked. The intuitive reason for this is that, with an implicit decryption model, only terms that have already been encrypted can be decrypted; one cannot "decrypt" other terms. This limitation masks possible attacks. We give an example of such an attack, but then we go on to show that a mild restriction on the protocol specification is enough to ensure that a free algebra with implicit decryption will find all attacks that a decrypt-extended algebra can find.

Our method of proof applies to symmetric encryption, but does not apply to public key encryption, although an analogous result may be possible. There are some observations in the Conclusion about the public key case.

Consider the protocol below, presented as a pair of strands. Assume that s is a secret, k is a secret key shared by A and B (but not the intruder), and c is a key known to both B and the intruder. X is a variable.



If X is instantiated with $\mathrm{d}(s, c)$, and the usual cancellation rule is applied, the message received by B becomes identical to the message sent by A, and the two strands form a *bundle*, in which every received message was either sent by a legitimate party or constructed by the intruder from previously sent messages. This bundle effectively compromises the secret s, since the intruder can encrypt X with c, yielding $\mathrm{e}(X, c) = \mathrm{e}(\mathrm{d}(s, c), c) = s$.

It is not possible to discover that s is compromised in the context of a free term algebra with no explicit decryption operator. This can be established using the technique in [MS01]. An informal argument is the following: an intruder cannot synthesize the term to be received by B, because it is encrypted using k, which is not known to the intruder. So the only possible source of the term is the transmitted message from A, namely, $\mathrm{e}(s, k)$. To recognize this message as the message received by B requires unifying the constant s with the term $\mathrm{e}(X, c)$, and this is impossible in the free algebra.

Essentially the same example also works for public key encryption. If we replace e by pe in the example, and assume that c is the public key of the intruder, we can let $X = \mathrm{pe}(s, c')$, where c' is the corresponding private key. The argument that s is not compromised in the free algebra is similar, but must be inductive, because the intruder can synthesize terms of the form $\mathrm{pe}(\mathrm{pe}(X, c), k)$. Such terms turn out not to be useful because the intruder must have X to make them.

Our objective is to show that, with an additional restriction, the free-algebra analysis will find all attacks that can be found using an extended algebra with decryption and cancellation. The additional restriction, called **EV-freedom**, is that there must be *no applications of an encryption operator to a variable (by itself) in the protocol specification.* That is, no message may be, or contain as a subterm, a term of the form $\mathrm{e}(X, k)$, where X is a variable and k is any term used as a key.

In particular, if the example above is modified to replace the X in B's received term by a pair $[X, a]$, for some constant a, the protocol is EV-free and secure, at least in the symmetric-key version. The argument depends on the ability to distinguish a pre-chosen constant s from an irreducible term of the form $\mathrm{e}([X, a], c)$, or equivalently, distinguishing $\mathrm{d}(s, c)$ from $[X, a]$. This is reasonable because the right-hand portion of the former value is not likely to match an independently chosen recognizable value a, as long as a is not too small. Furthermore, it is considered good protocol design practice to require that the format of received messages be verifiable through some known structural property of the data.

2 Protocol Model

Protocol roles are specified with strand schemas, or parametric strands, in which message terms may contain variables. Role specification with parametric strands was used in [SBP01] and [CDL00]. A *semibundle* (terminology from [SBP01]) is a collection of strand instances that might contain variables and which might not satisfy the condition that each received message must have been sent.

A semibundle is *solvable* if there is some bundle in which a ground instance of the semibundle is embedded. A secrecy goal can be phrased as a solvability problem for a semibundle containing a "test" strand in which a secret message is received (unencrypted). Some other security goals can be phrased as solvability problems as well.

Our result is stated in the context of two term algebras. The free algebra PSE contains a pairing operator $[X, Y]$ and a symmetric encryption operator $\mathrm{e}(X, K)$. The extended algebra, PSED, includes the decryption operator and two cancellation relations: $\mathrm{d}(\mathrm{e}(X, K), K) = X$ and $\mathrm{e}(\mathrm{d}(X, K), K) = X$. These relations are applied automatically as reduction rules (from left to right), and they yield a unique irreducible form for any term. (The fact that this term-rewriting system is convergent was proved in [Mea92].)

The solvability problem also requires us to specify the intruder capability, which we do with the free derivation rules below rather than with penetrator strands.

(A1a) $[X, Y] \vdash X$ (A2) $X, Y \vdash [X, Y]$ (A4) $X, K \vdash \mathrm{e}(X, K)$
(A1b) $[X, Y] \vdash Y$ (A3) $\mathrm{e}(X, K), K \vdash X$

We refer to (A1a) and (A1b) collectively as (A1). In the context of the decrypt-extended algebra there are two additional derivation rules:

(D1) $\mathrm{d}(X, K), K \vdash X$ (D2) $X, K \vdash \mathrm{d}(X, K)$

A ground term (a term containing no variables) t is *derivable* from a set of ground terms T if t is an element of T, or if there is some tree of derivation rule applications that generates t from elements of T. The nodes of a *derivation tree* for t are ground terms, the root of the derivation tree is t, the leaves are elements of T, and the parents of each term are the antecedents of the term in an instantiated rule. The set of terms derivable from T using only the free derivation rules is the *free* closure $\mathcal{C}(T)$, and the set of terms derivable from T using both the free and extended rules is the *decrypt-extended* closure $\mathcal{C}^D(T)$.

The approach to determining solvability in [MS01] is to note that any solution bundle, being a partial ordering, is consistent with at least one linear ordering of nodes in the semibundle, representing a possible temporal order of node events. For each linear node ordering, there is a *constraint set* consisting of the sequence of relations $R_i \in \mathcal{C}^D(S_i)$, where R_i is the i^{th} received message and S_i is the set of sent messages from nodes that precede the R_i node in the ordering, plus any initially known ground terms. We always have $S_i \subseteq S_{i+1}$.

3 The Free Closure Theorem

The main result of this paper is to show that if the protocol specification does not use the decrypt operator, so that it can be specified in PSE, and if it is EV-free, then every solution of a constraint set in PSED using the extended closure can be replaced by a solution using the free closure in PSE.

Let us say that a term t in PSED is *pure* if it contains no occurrence of the decrypt operator, so that it is also in PSE, otherwise it is called *impure*. A set of terms is pure if its elements are all pure. If a term t is described as impure, we mean that it is impure in its normal form after the cancellation reductions are applied, so that $\mathrm{d}(\mathrm{e}(u,k),k)$ reduces to u and is considered "pure" if u is pure. Variables are pure. A key result showing the motivation for EV-freedom is the following.

Lemma (Reduction). Suppose u is a pure term and σ is a substitution such that σu is reducible. Then u has a subterm $\mathrm{e}(X,y)$ where X is a variable such that $\sigma X = \mathrm{d}(v,w)$ for some terms v, w.

Proof. For cancellation reduction to be possible, σu must contain a subterm of one of the two forms (1) $\mathrm{d}(\mathrm{e}(v,w),w)$ or (2) $\mathrm{e}(\mathrm{d}(v,w),w)$. In case (1), since u is pure, the entire subterm must occur as a subterm of the image σX of some variable X. However, the image σX should be an irreducible ground term, so this case is not possible.

In case (2), as in case (1), the entire subterm cannot occur in σX. However, the remaining possibility is that u contains $\mathrm{e}(X,y)$ and $\sigma X = \mathrm{d}(v,w)$, where y is some term such that $\sigma y = w$. ■

Corollary (Reduction). EV-freedom of a pure term prevents reductions after substitution.

For any term t in PSED, let t^* be the term that results from changing every "d" operator in t to "e" at any depth. That is, $(\mathrm{d}(u,v))^* = \mathrm{e}(u^*,v^*)$, and for other functions f, $(f(u))^* = f(u^*)$. The star has no effect on constants or variables. We remark that:

Proposition (Star). For any t in PSED, t^* is pure; and if t is pure, $t^* = t$. ■

For any substitution σ, let σ^* map each variable X to $\sigma^* X = (\sigma X)^*$. We show next that in the presence of EV-freedom, the star operator can be applied through a substitution.

Lemma (Star Substitution). If u is pure and EV-free then $(\sigma u)^* = \sigma^* u$.

Proof. By induction on the structure of u. If u is a constant, neither σ nor $*$ has any effect. If u is a variable, $\sigma^* u$ is by definition $(\sigma u)^*$.

For larger structures, note that by the Reduction Corollary, no reductions are possible after a substitution by σ. This means that $\sigma f(v, w) = f(\sigma v, \sigma w)$, for any (non-d) function f, and this is true also for σ^*, because u is pure.

Suppose $u = f(v, w)$, where f is not "d" because u is pure, and note that the induction hypothesis may be applied to v and w. Then, $(\sigma u)^* = (f(\sigma v, \sigma w))^* = f(\sigma^* v, \sigma^* w) = \sigma^* f(v, w) = \sigma^* u$. ■

Now consider a constraint $u \in \mathcal{C}^D(T)$ that is satisfied by σ, so that there is a derivation tree deriving σu from σT. We are assuming that u and T are pure, but impure terms may be introduced in the derivation tree via σ or rule (D2). We plan to show that if $u \in \mathcal{C}^D(T)$ is satisfied by σ, then $u \in \mathcal{C}(T)$ is satisfied by σ^*. In particular, we can use the same derivation tree. This is possible because any use of the extended rule (D1) turns into an application of (A3), and any use of the extended rule (D2) turns into an application of (A4). For this to work, we must first check that every node in the tree, that is, any term t derivable from σT, may be replaced by t^*.

Theorem (Free Closure). Let T be pure and suppose $t \in \mathcal{C}^D(\sigma T)$. Then $t^* \in \mathcal{C}(\sigma^* T)$.

Proof. The proof is by induction on the size of the derivation tree. In the base case, $t \in \sigma T$. So there is a pure term $u \in T$ such that $\sigma u = t$. By the Star Substitution Lemma, $t^* = (\sigma u)^* = \sigma^* u \in \sigma^* T$.

Now consider a term t arising from a derivation rule whose antecedents satisfy the Lemma. That is, if s is an antecedent of t, $s^* \in \mathcal{C}(\sigma^* T)$. There are six cases, depending on which rule was used to derive t.

(A1). Assume that the derivation was $[t, s] \vdash t$. (The A1b case is similar.) By induction, $[t, s]^* \in \mathcal{C}(\sigma^* T)$. But $[t, s]^* = [t^*, s^*]$, so (A1a) can derive t^*.

(A2). $t = [r, s]$ and both $r^*, s^* \in \mathcal{C}(\sigma^* T)$. Then (A2) may be used to derive $t^* = [r, s]^* = [r^*, s^*]$.

(A3). Here, the derivation was $\mathrm{e}(t, k), k \vdash t$. Then $\mathrm{e}(t^*, k^*)$ and k^* are each in $\mathcal{C}(\sigma^* T)$. So (A3) may also be used here to derive t^*.

(A4). $t = \mathrm{e}(r, s)$ and both $r^*, s^* \in \mathcal{C}(\sigma^* T)$. Then $t^* = \mathrm{e}(r^*, s^*)$ and t^* is derivable using (A4).

(D1). Here, the derivation was $\mathrm{d}(t, k), k \vdash t$. Then $\mathrm{e}(t^*, k^*)$ and k^* are each in $\mathcal{C}(\sigma^* T)$. So (A3) may be used in this case also to derive t^*.

(D2). $t = \mathrm{d}(r, s)$ and both $r^*, s^* \in \mathcal{C}(\sigma^* T)$. Then $t^* = \mathrm{e}(r^*, s^*)$ and t^* is derivable using (A4). ■

Finally, given any of the constraints $u \in \mathcal{C}(T)$, if there is a solution $\sigma u \in \mathcal{C}^D(\sigma T)$, we conclude from the Free Closure Theorem that $(\sigma u)^* \in \mathcal{C}(\sigma^* T)$. But if u is pure and EV-free, $(\sigma u)^* = \sigma^* u$ and this tells us that σ^* is a solution of the constraint set over PSE. This yields a corollary:

Corollary (Free Closure). If a protocol strand-schema specification is expressed in PSE and is EV-free, then the corresponding constraint set is solvable over PSED with the extended closure if and only if it is solvable over PSE with the free closure.

4 Conclusion

We presented an example showing that cryptographic protocol analysis with a free term algebra and implicit decryption rules is inadequate in general, because it fails to find an attack involving explicit decryption. This is true even when the protocol can be specified in the free algebra. But if the protocol specification as a strand schema is EV-free, any attack found with the decrypt-extended algebra is also found with the free algebra.

EV-freedom is a reasonable requirement that is satisfied by most protocols. Note that it is not a requirement for security, but just a condition that makes analysis easier. EV-freedom is reminiscent of various tagging schemes that have been suggested to provide semantic security [GM84] and to prevent type flaw attacks based on data type ambiguity [HLS00]. These tagging schemes often yield EV-freedom as a byproduct.

The PSE algebra, with pairing and symmetric encryption, could be extended by including an abstract hash function or by assuming that pairing is associative, yielding n-ary concatenation, without affecting the proof approach.

Public-key encryption can be added to a free algebra as long as the private key corresponding to each public key does not appear explicitly. However, the approach used in this note does not appear to apply to public key encryption. One can extend the star operator by mapping each private key to its corresponding public key, but a problem arises in the proof of the Free Closure Theorem. In the public-key case corresponding to (D1), the derivation would be $\mathrm{pe}(t, k'_A) \vdash t$, where k'_A is a private key. The star version would then be $\mathrm{pe}(t^*, k_A) \vdash t^*$, which is valid only when k_A is the public key of the intruder. Although other private keys k'_A do not normally occur, they might arise from the substitution σ. An argument that such a substitution is unnecessary, or an argument taking a different approach, is still possible, and the validity of the result for public key encryption is considered open.

References

[CDL00] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Relating strands and multiset rewriting for security protocol analysis. In *13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society, 2000.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[HLS00] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *13th IEEE Computer Security Foundations Workshop*, pages 255–268. IEEE Computer Society, 2000.

[Mea92] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1/2):5–36, 1992.

[MS01] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communication Security*, pages 166–175. ACM SIGSAC, November 2001.

[Pau98] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[SBP01] D. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1):47–74, 2001.

[THG98] J. Thayer, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *11th IEEE Computer Security Foundations Workshop*, pages 66–78. IEEE Computer Society, 1998.

[THG99] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.