Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 125 (2005) 43–54
www.elsevier.com/locate/entcs

On the Relative Soundness of the Free Algebra Model for Public Key Encryption

Christopher Lynch[1,2]

Department of Mathematics and Computer Science
Clarkson Universtity
Potsdam, NY 13699-5815

Catherine Meadows[3]

Naval Research Laboratory
Center for High Assurance Computer Systems
Code 5543
Washington, DC 20375

Abstract

Formal systems for cryptographic protocol analysis typically model cryptosystems in terms of free algebras. Modeling the behavior of a cryptosystem in terms of rewrite rules is more expressive, however, and there are some attacks that can only be discovered when rewrite rules are used. But free algebras are more efficient, and appear to be sound for "most" protocols. In [9] Millen formalizes this intuition for shared key cryptography and provides conditions under which it holds; that is, conditions under which security for a free algebra version of the protocol implies security of the version using rewrite rules. Moreover, these conditions fit well with accepted best practice for protocol design. However, he left public key cryptography as an open problem. In this paper, we show how Millen's approach can be extended to public key cryptography, giving conditions under which security for the free algebra model implies security for the rewrite rule model. As in the case for shared key cryptography, our conditions correspond to standard best practice for protocol design.

Keywords: Cryptographic Protocol Verification.

[1] This work was done while the author was visiting the Naval Research Laboratory
[2] Email: clynch@clarkson.edu
[3] Email: meadows@itd.nrl.navy.mil

doi:10.1016/j.entcs.2004.05.018

1 Introduction

In most formal systems for cryptographic protocol analysis, it is typical to represent encryption operations in terms of a free algebra. Encryption, for example, may be represented as $e(K,X)$, where K is the key and X is the plaintext. Decryption, however, is represented implicitly. For example, one may include a rule that says that if a principal knows the shared/symmetric key K and $e(K,X)$, then it can also learn X.

Another approach is to represent both encryption and decryption explicitly. Encryption is represented, say, by $e(K,X)$, while decryption is represented by $d(K,X)$. The fact that decryption undoes encryption can be represented by a cancellation rule $d(K,e(K,X)) = X$. Since the decryption operator can also be used on unencrypted data, and the operations also cancel out when applied in reverse, we can also include the rule $e(K,d(K,X)) = X$.

This explicitness leads to a greater expressiveness, both in protocol representation and in constructing attacks. For example, in [5] Meadows analyzes a protocol that could not have been specified in the free algebra model, since it requires the application of the decryption operator to unencrypted data. Even when protocols do not require explicit representation of the decryption operator in this way, it is still possible that they may be subject to attacks that require its use. For example, in [9] Millen gives an example of a protocol that is secure in the free algebra model but insecure in the decrypt-extended model.

However, explicit representation of cancellation, although more expressive, can lead to a less efficient analysis. Moreover, many practical protocols appear to be immune to cancellation based attacks. Techniques such as recognizable formatting, probabilistic encryption, and so forth, appear to make unanticipated application of cancellation rules unlikely. Thus, most formal systems for cryptographic protocol analysis make use of the free algebra model, although they provide little formal justification for that approach.

In [9] Millen addresses this problem by giving necessary and sufficient conditions on protocols using symmetric encryption, under which if the secrecy property holds for a protocol in the free algebra model, then it also holds in the decrypt-extended model. The condition is specified on protocols written in a model based on parametric strand spaces and pattern matching. It has two subconditions. The first, called *purity* says that the decryption operator should not appear in the protocol specification. The second, called *EV-freeness*, says that there must be no application of an encryption operator to a variable in a specification. The EV-free property essentially says that a principal should not apply an encryption operator to a term unless it has been able to verify that

that term has some kind of structure, so that it is not unknowingly applying the encryption operator to the result of applying a decryption operator. Since it is generally considered good practice to apply encryption only to data about which one knows something (e.g. it is data that satisfies certain formatting conventions, or data that one has created oneself), it is reasonable to expect that most cryptographic protocols will satisfy EV-freeness.

Although Millen conjectured that similar results might hold for public key cryptography, he left it as an open question. As in shared key cryptography, it is possible to construct protocols vulnerable to attacks that rely on cancellation rules. Consider for example, a cryptosystem such as RSA in which signing and decryption are the same operation. In that case we have the identity $pke(privkey(U), pke(pubkey(U), X)) = X$, where pke is public encryption, $pubkey(U)$ is U's public encryption key, and $privkey(U)$ is both U's private signature key and its private decryption key. Consider now a simple notary public protocol in which a server is willing to sign any message it gets. Suppose that someone encrypts a message m with the server's public key and sends it. An intruder can send $pke(pubkey(s), m)$ to the server, which will output $pke(privkey(s), pke(pubkey(s), m)) = m$.

Such an attack relies on the cancellation properties of public key encryption, and can't be represented without it. But it is also the case that most practical protocols using public key cryptography are immune to this type of attack. Typically, public key pairs for signing are different from the pairs used for encryption, and digital signatures are applied to a one-way hash of a message, not the message itself. Finally, it is considered sound procedure never to sign a message that is entirely supplied by another party. Instead, a principal is usually required to add some material of its own before signing.

Such observations led us to believe that it should be possible to extend Millen's results to public key cryptography, basing our restrictions on the commonly accepted design principles described above. Our first, public key purity, which is analogous to Millen's purity, is that different key pairs should be used for encryption/decryption and signing/verification, even when the same algorithm is used for both signing and encryption, and that the private decryption key and the public signature verification key never appear in the protocol specification. The second of these, analogous to EV-freedom, called PEV-freedom, is that no occurrence of an encryption or signing operator contains a variable as an argument. We also introduce a third condition, called structure, which is sufficient without the other two, which says that no occurrence of an encryption or signing operator contains as an argument a variable or a term rooted with a public key operator. Note that the requirement that different key pairs be used for different types of operations assumes that prin-

cipals know which key pairs are intended for what, and so rules out protocols that include the distribution of public keys. This is not the case when we require structure only, however.

The rest of the paper is organized as follows. In Section 2 we introduce the model. In Section 3 we prove the soundness results for public key pure, PEV-free protocols. In Section 4 we describe our plans for further work and make some conjectures. Due to page limits, we were unable to include much of the technical material for this paper. All of the proofs were left out, and all of the material on *structured protocols* was left out. The full paper with all of that material can be found at `www.clarkson.edu/~clynch/papers/pubfree.ps/`

2 Protocol Model

2.1 Millen and Shmatikov's Constraint Model and its Application to the Free Algebra Model in Single Key Cryptography

Like Millen, we use a model based on parametric strands as used by Cervesato et al. in [1] and Song et al. in [12], and constraint solving as developed by Millen and Shmatikov in [10]. We refer the reader to [3] for a definition of strand spaces. In a parametric strand, message terms may contain variables. A variable will correspond to a subterm of a message term appearing in a negative node for which the receiver can verify no properties; thus any terms can be used to substitute for it. Variables can also appear in positive nodes of a strand, but only if they appeared earlier in a negative node.

Definition 2.1 A *constraint* is a pair $m : T$ where m is a term and T is a term set. A set of constraints $\mathcal{C}$ is *satisfied* by a substitution σ if, for all $m : T \in \mathcal{C}$, the intruder can derive $m\sigma$ from $T\sigma$.

A set of constraints and term sets can be constructed from a sequence of positive and negative nodes in a semibundle $\mathcal{B}$ that is compatible with the partial order imposed by the bundle ordering. Each positive node expands the last term set by the message it contains, while each negative node creates a constraint $m : T$ where m is the message in the node and T is the last term set, and the first term set is the set of terms known initially by the intruder. Such a constraint set is satisfied by a substitution σ, if and only if, for each initial sequence s of the constraint immediately preceding a negative node m, the intruder can generate $m\sigma$ from the positive nodes in $s\sigma$. In other words, the constraint set is satisfied by σ if and only if the intruder can generate a message expected by an honest principal from all the previous messages sent by honest principals and all the information known initially by the intruder. This, of course, is exactly how we use strand spaces to construct an attack on

a protocol.

This makes it possible to define a bundle in a sense compatible with constraints.

Definition 2.2 A *semibundle* is a set of parametric strands. A total order $<$ on nodes in a semibundle is *compatible* with the semibundle if whenever N_1 precedes N_2 on a strand, then $N_1 < N_2$.

Definition 2.3 A *bundle* is a semibundle B such that there exists a total order $<$ on nodes compatible with B such that the set of constraints produced by the sequence of nodes induced by $<$ is satisfied. A semibundle is *solvable* if there is some bundle in which some ground instance of the semibundle is embedded.

In [9] Millen describes two constraint systems. One is based on a free algebra, with intruder operators consisting of concatenation, deconcatenation, encryption, and reversal of encryption. The other is based on an algebra (called the *decrypt-extended* algebra) that includes cancellation rules describing the effects of encryption and decryption on each other, and penetrator operators consisting of concatenation, deconcatenation, encryption, reversal of encryption, and the application of the decryption operator. The main theorem of that paper is that, given certain conditions, any bundle that is solvable in the second system is solvable in the first system. This boils down to showing under what conditions a constraint $m : T$ from the free algebra that is solvable in the cancellation algebra is also solvable in the free algebra. In [9] these conditions are that the protocol from which the constraints are derived should not contain any explicit use of the decryption operator (purity), and that it should contain no case of an encryption operator applied to a variable (EV-freeness). Note that solvability in the free algebra always gives us solvability in the decrypt-extended algebra, so that the result is an if-and-only-if. Moreover, the solution for the decrypt-extended algebra induced by the one for the free algebra introduces no strands belonging to honest principals. This has a result on the type of theorems we can prove, as we will see below.

Millen describes in [9] how this result can be used for proving that the free algebra is sound with respect to the decrypt-extended algebra for secrecy properties. One can create a special strand that describes the intruder learning a specified secret, such as a key. One can use his theorem to show that purity and EV-freeness implies that the intruder strand is solvable in the cancellation algebra if and only if it is solvable in the free algebra. Millen also mentions that the result could be used to prove soundness with respect to some authentication properties, although he does not go into detail about this.

We would like to characterize the types of attacks for which Millen's result and our results apply. To this end, we define a specification of an attack below.

Definition 2.4 We define an *attack specification* to be two sets of parametric strands, called the positive and the negative set. A *ground attack specification* is an attack specification in which all terms are ground.

Briefly, the positive set of an attack specification gives the strands that should be in the bundle, and the negative set gives the strands that should not be in the bundle. Thus, a specification of an attack on the secrecy of a key accepted by an initiator could be given in terms of a positive set containing two strands: an initiator strand in which the key is represented by the variable K, and a special intruder strand of the sort described by Millen in which the datum learned by the intruder is also represented by K. The negative set would be empty. However, a specification of an authentication protocol would include the authenticated strand in the positive set and the authenticating strand in the negative set.

The notion of attack specification is not explicitly set out in [10], but our definition captures the essence of the kinds of attacks Millen and Shmatikov handle in their constraint system. We note, however, that Millen and Shmatikov restrict themselves to ground attack specifications, or at least attack specifications in which all terms in the positive set are ground. We believe that it should be straightforward to extend their notion of an attack to non-ground attack specifications. This would allow us to characterize attacks in a somewhat more general way. However, when rewrite rules are used to describe the properties of a cryptosystem, the problem becomes a little more complex, since we need to keep track of which terms in an attack specification are assumed to be irreducible, and which terms may be allowed to be reducible. Though this would not necessarily be difficult to do (something like this is already done for the specification of insecure states in the NRL Protocol Analyzer) it could be tedious and somewhat beside the main goal of this paper. Thus we treat it as outside the scope of the paper and leave it for further work.

Millen's result, and our result as well, extends easily to any grounded attack specification with an empty negative set. It may not always extend to the case of a non-empty negative set, however, since the transformation that maps the cancellation algebra to the free algebra is not necessarily the identity, and it is possible that a strand that appears in an attack in the free algebra could be sent into the negative set by the transformation.

2.2 Free and Extended Algebras for Public Key Cryptography

As in Millen's case, our result is stated in terms of two term algebras. These are defined as follows:

Definition 2.5 The free algebra *FA* contains

(i) constants *pub*, *priv*, *enc* and *sig*. It also contains a set of names, a set of messages, and a set of nonces.

(ii) a pairing operator $[X, Y]$ and a public key encryption operator $pe(X, Y)$. The first argument to *pe* is the key, which we represent as $pk(N, P, S)$, where
- N is the name of the key (usually the principal who uses this key),
- P can be either *pub* or *priv*, indicating whether this is a public key or a private key, and
- S can be either *enc* or *sig*, indicating whether the key is used for encryption or for signing.

The second argument to *pe* is the message.

For example, the term $pe(pk(A, pub, enc), pe(pk(B, priv, sig), m))$ represents message m, signed with B's private key and then encrypted with A's public key. We use this notation to enforce the assumption that no key will be used for both signing and encryption in a protocol.

We only consider protocols where the second and third arguments of a *pk* operator are not variables. This reflects the assumption that keys have a fixed purpose, it is not possible to change that purpose, and all principals will recognize what the purpose of a key is. This would hold, for example, in cases in which public and private keys are distributed beforehand by some trusted protocol. Since this is an assumption under which much cryptographic protocol analysis is done, we consider it reasonable, at least as a first approximation.

Definition 2.6 The extended algebra EFA, besides the above, includes the following set of equations E:

E1 $pe(pk(K, pub, enc), pe(pk(K, priv, enc), X)) = X$

E2 $pe(pk(K, priv, enc), pe(pk(K, pub, enc), X)) = X$

E3 $pe(pk(K, pub, sig), pe(pk(K, priv, sig), X)) = X$

E4 $pe(pk(K, priv, sig), pe(pk(K, pub, sig), X)) = X$

This set E models the properties of the encryption and signature operations. Let R be the rewrite system which results from orienting these equations from left to right as rewrite rules. This can be shown to be a confluent and terminating rewriting system representing E, using techniques similar to those

in [5]. This means that two terms are equivalent modulo E if and only if they have the same unique normal form modulo R. We define $t\downarrow_R$ to be the normal form of t modulo R.

Let **D** be the following derivation rules:

D1 $(X, Y) \vdash X$

D2 $(X, Y) \vdash Y$

D3 $X, Y \vdash (X, Y)$

D4 $pe(pk(K, pub, enc), X), pk(K, priv, enc) \vdash X$

D5 $pe(pk(K, priv, sig), X), pk(K, pub, sig) \vdash X$

D6 $X, pk(K, pub, enc) \vdash pe(pk(K, pub, enc), X)$

D7 $X, pk(K, priv, sig) \vdash pe(pk(K, priv, sig), X)$

The list **D** expresses the ways in which the intruder can derive information in the free algebra FA.

We next construct a set of derivation rules **DE** which expresses what the intruder can learn in the equational extension EFA of the free algebra. We let **DE** be the set containing the derivation rules of **D**, not including rules **D4** and **D5**, plus the additional rules

DE1 $X, pk(K, priv, enc) \vdash pe(pk(K, priv, enc), X)$

DE2 $X, pk(K, pub, sig) \vdash pe(pk(K, pub, sig), X)$

If T is a set of terms, then we define $Deriv_{\mathbf{D}}(T)$ to be the set of terms that can be derived from T using the derivation rules of **D**. We define $Deriv_{\mathbf{DE}}(T)$ as the set of terms that can be derived from T using the derivation rules **DE**.

The derivations of $Deriv_{\mathbf{DE}}$ take place modulo the equational theory E. In other words, given a derivation rule, $t_1 \cdots t_n \vdash t$ and a set of terms $s_1 \cdots s_n$, then we can derive a new term s if $s = t\sigma$ and $s_i =_E t_i\sigma$ for all i with $1 \leq i \leq n$. We assume that the substitution σ used in a derivation is irreducible by R. (If it were not, it could always be replaced by its reduced form.) After each derivation rule, we reduce the new term s by R.

Note that these derivation rules are analogous to the derivation rules given by Millen[9] for his result for secret keys. We only differ from them in that in **DE**, we do not explicitly add a rule to say that if a privately encrypted message is known, and if the corresponding key is known, then the message is known. We do not add this because it is a consequence of a derivation rule and a reduction rule. Similarly for messages publicly signed.

Any term that can be derived by **D** can also be derived by **DE**. The only thing that is not obvious is for rules 4 and 5 of **D** that do not exist in **DE**. But these rules are simulated by a derivation of **DE** plus a reduction with R.

2.3 Syntactic Properties of Protocol Specifications

We now define some basic syntactic properties of protocols which we will use to prove our results.

Definition 2.7 We call a term t *structured* if whenever the operator pe appears in t, the second argument of pe is not a variable or rooted with the operator pe.

An example of a structured term would be a tagged message such as in [4], in which every term is preceded by a tag describing its type. Another example would be a pair of terms.

Definition 2.8 We call a term t *public key pure* (or pk-pure) if it does not contain any term $pk(k, priv, enc)$ or $pk(k, pub, sig)$ where k may be any term. A set of terms is *pk-pure* if all its elements are pk-pure. A protocol is considered to be *pk-pure* if all terms appearing in the protocol are pk-pure.

Note that all pk-pure terms are irreducible by R.

Definition 2.9 A protocol is considered to be *PEV-free* if no occurrence of pe contains a variable as an argument.

3 pk-Purity and PEV-freeness Imply Soundness

We will only consider pk-pure protocols in this section, because we wish to model real protocols where decryptions are implicit. Similarly, the recognition of signatures is also implicit.

We will show that, if a protocol is pk-pure and PEV-free, and if T is pk-pure, then we will show that $Deriv_{\mathbf{DE}}(T) \subseteq Deriv_{\mathbf{D}}(T)$, which implies that $Deriv_{\mathbf{DE}}(T) = Deriv_{\mathbf{D}}(T)$. This will imply that the free algebra model finds all attacks that the extended model does.

First we will need a lemma that says that if a pk-pure irreducible term is PEV-free, then it remains irreducible after applying an irreducible substitution.

Lemma 3.1 *Let t be a pk-pure PEV-free term, and σ be a substitution such that σ is irreducible by R. Then $t\sigma$ is irreducible by R.*

We show that all derivations in **DE** preserve certain properties. To do that, we define a new rewrite system P which reduces all terms to a pk-pure term.

(i) $pe(pk(K, priv, enc), X) \rightarrow X$

(ii) $pe(pk(K, pub, sig), X) \rightarrow X$

We define $t \downarrow_P$ to be the normal form of t modulo P. Note that every pk-pure term is irreducible modulo P.

Lemma 3.2 *Let t be a pk-pure term, and σ be a substitution. If σ is irreducible by P, then $t\sigma$ is irreducible by P.*

This implies that $(t\sigma) \downarrow_P = t(\sigma \downarrow_P)$ if t is pk-pure.

Theorem 3.3 *Let $s_1, \cdots, s_n$ be terms, irreducible by R, such that s can be derived in one step from $s_1, \cdots, s_n$ in* **DE**. *Then either $s \downarrow_P$ can be derived in one step from $s_1 \downarrow_P, \cdots, s_n \downarrow_P$ in* **D** *and s is irreducible by R, or there exists an i such that $s \downarrow_P = s_i \downarrow_P$.*

Theorem 3.4 *Suppose that T is a set of terms irreducible by R. Let t be a term. If $t \in Deriv_{\mathbf{DE}}(T)$ then $t \downarrow_P \in Deriv_{\mathbf{D}}(T \downarrow_P)$. In addition, there exists an $s \in Deriv_{\mathbf{DE}}(T)$ such that s is irreducible by R and $t \downarrow_P = s \downarrow_P$.*

Corollary 3.5 *Suppose that T is a set of PEV-free pk-pure terms. Let t be a pk-pure PEV-free ground term. Let σ be a substitution irreducible by R. If $t\sigma \in Deriv_{\mathbf{DE}}(T\sigma)$ then $t(\sigma \downarrow_P) \in Deriv_{\mathbf{D}}(T(\sigma \downarrow_P))$.*

Corollary 3.6 *Let Pr be a pk-pure PEV-free protocol. Let B be a semibundle of Pr. If B is a bundle in EFA, then it is a bundle in FA.*

4 Conclusions

In this paper we extended Millen's result on the soundness of the free encryption model for shared key encryption to public key encryption as well. Although the proof differed in some important ways from Millen's we found that his general approach extended quite well to public key cryptography. Moreover, we found that our results, similarly to Millen's corresponded well to certain well-established best practices in cryptographic protocol design.

There are a number of ways in which these results could be extended, besides to attack specifications with nonempty negative sets as we noted in the previous section. The most obvious would be to include other best practices (for example, the use of probabilistic encryption), or to extend our results to models in which both public and single key encryption are used, since most protocols in use today employ both. Moreover, there are a number of other special properties of cryptographic algorithms for which it would be useful to be able to abstract away from in a safe way. For example, the commutative property of Diffie-Hellman can add greatly to the expense of the analysis of a cryptographic protocol. In [7] a specification of Diffie-Hellman is used that abstracts away from the commutative property, but it has the disadvantage that the same key fragment computed by an initiator and a responder will have

different representations. This should not be a problem if we can guarantee that a key fragment computed by an initiator will never be confused with one computed by a responder. We have some preliminary results in that direction, using techniques similar to those used in this paper. Other algebraic properties of interest that might be amenable to techniques such as the ones used in [9] and here include the homomorphic properties of RSA used to compute blind signatures, as well as the properties of Diffie-Hellman used in Group Diffie-Hellman. Indeed, an early paper by Even et al. [2] shows that it is safe to leave out the homomorphism property for RSA for a very restricted class of protocols known as ping-pong protocols, so we know that for this property the result is true at least for a limited case. It would also be useful to extend these results to systems that already use cancellation rules, such as the NRL Protocol Analyzer [6]. This would allow us to use cancellation rules when necessary, and the more efficient free algebras when not.

The approach taken in both Millen's and our paper, as well as the proposed work described above, is part of a more general plan that we have outlined in [8]. The general goal is to have a hierarchy of protocol models, each containing different amounts of detail. Given certain conditions on a protocol, one can prove that a less detailed model is sound with respect to the more detailed model, and perform one's analysis with the more efficient but less detailed model. If the conditions are not satisfied, one works with the more detailed model. This approach is not limited to more algebraic properties of cryptosystems; other possibilities, detailed in [8] include type flaw attacks (handled by Heather at al. in [4]), the possibility of key compromise, and cryptographic models. We consider the results in this paper to be a potential building-block in that edifice.

5 Acknowledgments

This work was supported by the Office of Naval Research.

References

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

[2] S. Even, O. Goldreich, and A. Shamir. On the security of ping-pong protocols when implemented using the RSA. In *Advances in Cryptology: Proceedings of Crypto '85*. Springer-Verlag, 1985.

[3] F. Thayer Fábrega, J.. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 1998 IEEE Symposium on Security and Privacy*, pages 160–171. IEEE Computer Society Press, May 1998.

[4] J. Heather, S. Schneider, and G. Lowe. How to prevent type flaw attacks on security protocols. In *Proc. 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2000.

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

[6] C. Meadows. The NRL Protocol Analyzer: an overview. *Journal of Logic Programming*, 26(2):113–131, 1995.

[7] C. Meadows. Analysis of the Internet Key Exchange Protocol using the NRL Protocol Analyzer. In *Proc. 1999 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, June 1999.

[8] C. Meadows. Towards a hierarchy of cryptographic protocol specifications. In *Proc. FMSE 2003: Formal Methods in Security Engineering*. ACM Press, 2003.

[9] J. Millen. On the freedom of decryption. *Information Processing Letters*, 86(6):329–333, June 2003.

[10] J. Millen and V. Shmatikov. Constraint solving for bounded process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 166–175. ACM Press, 2001.

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

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