Equal To The Task?

James Heather[1] and Steve Schneider[2]

[1] Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, UK.
Email: j.heather@eim.surrey.ac.uk
[2] Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK. Email: steve@cs.rhul.ac.uk

Abstract. Many methods of analysing security protocols have been proposed, but most such methods rely on analysing a protocol running only a finite network. Some, however—notably, data independence, the strand spaces model, and the rank functions model—can be used to prove correctness of a protocol running on an unbounded network.
Roscoe and Broadfoot in [17] show how data independence techniques may be used to verify a security protocol running on an unbounded network. They also consider a weakness inherent in the RSA algorithm, discovered by Franklin and Reiter [3], and show that their data independence approach cannot deal with an intruder endowed with the ability to exploit this weakness.
In this paper, we show that neither can the use of honest ideals in the strand spaces model or the use of rank functions in the CSP model be easily adapted to cover such an intruder. In each case, the inequality tests required to model the new intruder cause problems when attempting to extend analysis of a finite network to cover an unbounded network. The results suggest that more work is needed on adapting the intruder model to allow for cryptographic attacks.
Key words: cryptographic protocols; rank functions; strand spaces; data independence; inequality tests; RSA; formal methods in security; security models; security verification.

1 Introduction

In [3], Franklin and Reiter present an attack on the RSA algorithm [15], allowing decryption of a pair of distinct encrypted messages, in the case where the two ciphertexts stand in a known linear relation and are encrypted under the same RSA public key with a public-key exponent of 3.

Suppose that we have two encrypted messages

$$c_1 = \{m\}_k$$
$$c_2 = \{am + b\}_k$$

and that we know a and b (with $a \neq 0$, and not both $a = 1$ and $b = 0$), and the public key k, but we do not know m. Suppose further that the exponent of k is

3, and that the modulus is n. Then we have

$$c_1 = m^3 \bmod n$$
$$c_2 = (am + b)^3 \bmod n$$

We can recover m by observing that

$$\begin{aligned} m \bmod n &= \frac{3a^3bm^3 + 3a^2b^2m^2 + 3ab^3m}{3a^3bm^2 + 3a^2b^2m + 3ab^3} \\ &= \frac{b(c_2 + 2a^3c_1 - b^3)}{a(c_2 - a^3c_1 + 2b^3)} \end{aligned}$$

Coppersmith et al. generalise this attack in [1] to deal with an arbitrary public-key exponent e, but the complexity of the attack is $O(e \log^2 e)$. They claim that the attack should be practical for exponents of up to 2^{32}. In particular, it is efficient with an exponent of $2^{16} + 1$, a popular choice in many applications.

This paper explores the consequences of this attack for security protocol analysis. Specifically, it investigates the feasibility of tailoring existing protocol verification methods to allow an intruder to take advantage of the RSA attack described above. For one of these methods—data independence—it has already been shown by Roscoe and Broadfoot that the attack cannot be incorporated into the model without destroying the foundation on which the model is built. We demonstrate in this paper that two other current methods—rank functions for CSP models, and honest ideals for strand spaces—have surprisingly similar problems in allowing for this strengthened intruder.

The structure of the paper is as follows. In the next section, we describe how an intruder might use the RSA attack to break a security protocol, and give a brief overview of how one might attempt to model this capability when analysing a protocol. In Section 3, we summarise Roscoe and Broadfoot's findings with regard to the RSA attack and the data independence model. Section 4 describes the rank functions model, and investigates the possibility of including the RSA attack among the rank functions intruder's weapons. Section 5 deals with the same question for the strand spaces model. Finally, in Section 6, we sum up and draw conclusions.

2 Exploiting the attack to break security protocols

If concatenations of smaller protocol messages are formed by simple concatenations of the bit strings representing those messages, then the concept of a linear relation between messages is clear: forming the compound message $x.y$ will involve multiplying the value of x by $2^{\ell(y)}$ and adding the value of y (where '$\ell(y)$' denotes the length of the bit string representing y). The most natural formulation of this is when $m_1 = a.X.b$ and $m_2 = c.X.d$, with the intruder already knowing a, b, c and d (any of which could be null, but not both $a = c$ and $b = d$). For then

$$m_2 = 2^{\ell(d)-\ell(b)}(m_1 - b - 2^{\ell(X.b)}a) + d + 2^{\ell(X.d)}c$$

An intruder who knows $\{m_1\}_k$ and $\{m_2\}_k$ for some known RSA key k with a low encrypting exponent can make use of the attack given above to deduce the value of X.

2.1 Modelling the new intruder

In each of the three models under consideration, the intruder has complete control over the entire network, in the style of the Dolev-Yao model [2]. The intruder may

- allow messages to be sent normally from agent to agent (but take note of the contents of the message in the process);
- intercept messages and fail to deliver them;
- construct and deliver spurious messages purporting to come from anyone he pleases.

In this last case, he may send any message that he has already seen in the network or that he can produce using only messages that he has seen. For instance, if he has observed N_A and N_B as separate messages, then he may construct $N_A.N_B$ from them and deliver this concatenation. (This concatenation operator is defined to be associative.)

To allow for the extra intruder capability of exploiting the RSA weakness when conducting protocol analysis, one has to introduce a new deduction rule, or type of penetrator strand, or the analogue in the model in question. With a rank functions approach, this will be an extra $\vdash$ rule (explained more fully in Section 4):

$$\{\{a.X.b\}_k, \{c.X.d\}_k, a, b, c, d\} \vdash X \qquad (a \neq c \text{ or } b \neq d)$$

In the strand spaces model we would extend the definition of a penetrator strand to include a new type **L** (see Section 5):

L ***Low-exp RSA*** $\langle -\{a.X.b\}_k, -\{c.X.d\}_k,$
$-k, -a, -b, -c, -d, +X\rangle$ for $a, b, c, d, X \in \mathsf{A}$; $a \neq c$ or $b \neq d$; and $k \in \mathsf{K}$.

The data independence CSP model would contain a new set of deductions along the following lines, describing how the intruder, already in possession of messages from a set Z, can extend his knowledge by utilising the RSA attack to deduce a new message f. The set $deductions(Z)$ is a set of ordered pairs of the form $\langle S, m\rangle$, corresponding to $S \vdash m$ in the rank functions world (Section 3):

$$\begin{aligned} deductions(Z) = \{\langle\{a, b, c, d, k, \{a.f.b\}_k, \{c.f.d\}_k\}, f\rangle \mid \\ \{a.f.b\}_k \in Z, \{b.f'.d\}_{k'} \in Z, f = f', k = k', a \neq c \vee b \neq d\} \end{aligned}$$

This definition differs from the one given in [17] in three respects, none of great moment:

1. The name given in [17] is '`deductions5`'.

2. The key k has been erroneously omitted in [17] from the set of messages required in order to make the deduction; it is here included.
3. Roscoe and Broadfoot give the deduction from $\{a.f\}_k$ and $\{b.f\}_k$, and leave the more general case (with c and d included) for another (implied) deduction rule. Here, we give the full case with $\{a.f.b\}_k$ and $\{c.f.d\}_k$, and assume that the model will be constructed so that a message of the form $a.f$ can be parsed as $a.f.b$ with $b = \langle\rangle$ (the null message—and we naturally further assume that an intruder can produce the null message whenever required).

Note the inequality tests in each case. Without them, the rule would degenerate: in the case where $a = b = c = d = \langle\rangle$, we would have rank functions deductions of the form

$$\{\{X\}_k\} \vdash X$$

and penetrator strands of the form

$$\langle -\{X\}_k, +X \rangle$$

and a subset of $deductions(Z)$ as

$$D(Z) = \{\langle\{\{f\}_k, k\}, f\rangle \mid \{f\}_k \in Z\}$$

In other words, the intruder would be able to break every encryption.

The remainder of this paper deals with the issues involved in attempting to include this new deduction rule or strand type in the analysis, and in particular with the effect on proving correctness of a security protocol running on an unbounded network.

3 Data independence

Roscoe and Broadfoot [17] have demonstrated how one may analyse a security protocol running on a finite network, and then use Lazić's work on data independence [11, 12] to extend the results to cover unbounded networks.

Broadly speaking, a CSP process P, parameterised by a data type T, is said to be *data independent* with respect to T if the operation of P does not depend in any way on T. The results that Lazić presents give conditions under which the abstract data type T may be replaced with a small, concrete data type T' without affecting the whether P has particular properties.

Roscoe and Broadfoot put forward the concept of a *positive deductive system*. A deductive system is positive relative to a type T essentially if it treats all members of T equivalently, and never requires inequality between two members of type T in the set from which the deduction is made. They then show that if the intruder model is built over a positive deductive system, it satisfies Lazić's *Positive Conjunctions of Equality Tests* (**PosConjEqT**) property (or a slight variant), which requires that a process should not perform explicit or implicit equality tests except for equality tests after which the process halts whenever

the test gives a negative result. For data independence techniques to apply to a model of a network running a security protocol, we need the process representing the intruder, and also every process representing an honest agent, to satisfy this property.

The authors use this result to replace infinite sets of nonces and keys with finite sets while ensuring that no attacks will be defeated by the replacement. Model checking can then be used to show that there are no attacks on a small system, and the data independence theorems provide the guarantee that there will consequently be no attacks on an unbounded system.

As Roscoe and Broadfoot make clear, however, giving the intruder the power to exploit the weakness in low-exponent RSA, by allowing him to make use of the set $deductions(Z)$ defined earlier, results in an intruder process that does not satisfy this condition. The intruder must check that he holds two distinct encryptions in order to perform the attack; and this check (the '$a \neq c \vee b \neq d$' in the definition of $deductions(Z)$) violates the **PosConjEqT** condition.

The necessity of the inequality check in specifying the RSA attack makes it impossible to work this new intruder capability into the data independence model without invalidating the results.

4 Rank functions

In this section, we describe the rank functions approach to security protocol analysis, as set forth in [18, 9] and discuss the consequences of attempting to strengthen the rank functions intruder model to endow him with the ability to use the attack on low-exponent RSA.

Some knowledge of CSP is assumed. For an introduction to CSP, see [10, 16, 19].

4.1 The network

The network considered in [9] consists of a (usually infinite) number of honest agents, and one dishonest enemy. The behaviour of an agent J is described as two CSP process U_J^I and U_J^R, controlling respectively J's actions as initiator and responder. These user processes will vary according to the protocol under consideration; they will consist of communication along channels *trans*, representing transmission of a message, and *rec*, representing reception.

We shall write '$\mathcal{U}$' for the set of user identities on the network (including identities of any dishonest agents), and '$\mathcal{N}$' for the set of nonces that can be used in protocol runs.

We define a 'generates' relation $\vdash$, writing '$S \vdash m$' to denote that the enemy may construct message m if he possesses every message in the set S. If m and n are messages, k is a key, and k^{-1} is the inverse of k, then $\vdash$ is the smallest

relation that satisfies

$$\begin{aligned} \{m, n\} &\vdash m.n \\ \{m.n\} &\vdash m \\ \{m.n\} &\vdash n \\ \{m, k\} &\vdash \{m\}_k \\ \{\{m\}_k, k^{-1}\} &\vdash m \end{aligned}$$

and also satisfies the closure conditions

$$\begin{aligned} m \in S &\Rightarrow S \vdash m \\ (\forall\, v \in T \bullet S \vdash v) \wedge T \vdash m &\Rightarrow S \vdash m \end{aligned}$$

The enemy (already in possession of a set of messages S) is then described by the recursive definition:

$$ENEMY\,(S) =$$

$$trans?i?j?m \rightarrow ENEMY\,(S \cup \{m\}) \;\Box \;\Box_{i,j,S \vdash m}\; rec!i!j!m \rightarrow ENEMY\,(S)$$

Here the enemy can either receive any message m transmitted by any agent i to any other agent j along a *trans* channel, and then act as the enemy with that additional message; or pass any message m that it can generate from S to any agent i along its *rec* channel, remaining with the same information S.

The whole network is then

$$\mathbf{NET} = (\;|||_{J \in \mathcal{U}}\; (U_J^I \;|||\; U_J^R)) \;\|\; \mathbf{ENEMY}$$

For any given protocol, there will be a (possibly infinite) set of all atoms that could ever appear in a message of the protocol. This set will encompass all the user identities, nonces and keys, and any other types of atom used in the protocol (for instance, timestamps). From this set we can construct the *message space*, usually denoted by '$\mathcal{M}$', which is the space of all messages that can be generated from these atoms.

We use '*INIT*' to denote the set of atoms known to the enemy right from the start. Some users will be under the control of the enemy, and hence their secret keys and all nonces that they might produce will be in *INIT*; other users will be free of enemy control, and so their secret keys and nonces will not be in *INIT*.

4.2 Authentication

For an authentication protocol to be correct, we usually require that a user B should not finish running the protocol believing that he has been running with a user A unless A also believes that he has been running the protocol with B. (For a discussion of different forms of authentication, see [18].) Conditions such

as this can easily be expressed as trace specifications on **NET**, requiring that no event from a set T has occurred unless another event from a set R has previously occurred.

Definition 1. *For sets $R, T \in \mathcal{M}$, we define the trace specification R* **precedes** *T as*

$$P \textbf{ sat } R \textbf{ precedes } T \Leftrightarrow \forall\, tr \in traces(P) \bullet (tr \upharpoonright R = \langle\rangle \Rightarrow tr \upharpoonright T = \langle\rangle)$$

and note that, since all processes are prefix-closed, this guarantees that any occurrence of $t \in T$ in a trace will be preceded by an occurrence of some $r \in R$.

4.3 Rank functions

Definition 2. *A rank function, as defined in [7], is a function*

$$\rho : \mathcal{M} \to \{0, 1\}$$

from the message space to the binary-valued set $\{0, 1\}$. In addition, we define

$$\mathcal{M}_{\rho^-} = \{m \in \mathcal{M} \bullet \rho(m) = 0\}$$
$$\mathcal{M}_{\rho^+} = \{m \in \mathcal{M} \bullet \rho(m) = 1\}$$

The point of a rank function is to partition the message space into those messages that the enemy might be able to get hold of, and those messages that will certainly remain out of his grasp. Anything with a rank of one will be something that the enemy might get his hands on; anything of zero rank will be unavailable to him.

4.4 The central theorem from [9]

For a process P to *maintain the rank* with respect to a rank function ρ, we mean that it will never transmit any message m with $\rho(m) = 0$ unless it has previously received a message m' with $\rho(m') = 0$. Essentially, this means that the process will never give out anything secret unless it has already received a secret message.

Definition 3. *We say that P* **maintains** *ρ if*

$$P \textbf{ sat } rec.\mathcal{U}.\mathcal{U}.\mathcal{M}_{\rho^-} \textbf{ precedes } trans.\mathcal{U}.\mathcal{U}.\mathcal{M}_{\rho^-}$$

Theorem 1. *If, for sets R and T, there exists a rank function $\rho : \mathcal{M} \to \{0, 1\}$ satisfying*

1. $\forall\, m \in INIT \bullet \rho(m) = 1$
2. $\forall\, S \subseteq \mathcal{M}, m \in \mathcal{M} \bullet ((\forall\, m' \in S \bullet \rho(m') = 1) \wedge S \vdash m) \Rightarrow \rho(m) = 1$
3. $\forall\, t \in T \bullet \rho(t) = 0$
4. $\forall\, J \in \mathcal{U} \bullet USER_J \ |[\, R \,]|\ Stop$ **maintains** ρ

then ***NET*** **sat** R **precedes** T.

The proof is omitted; the interested reader is advised to consult [18, 9]. For a demonstration of how Theorem 1 can be used to verify security protocols, see [9].

4.5 Strengthening the intruder model

The following protocol is based on Lowe's fixed version [13] of the Needham-Schroeder Public Key Protocol [14]:

Msg 1. $a \to b \ : \ \{a.na\}_{PK(b)}$
Msg 2. $b \to a \ : \ \{na.nb.b\}_{PK(a)}$
Msg 3. $a \to b \ : \ \{nb\}_{PK(b)}$
Msg 4. $b \to a \ : \ nb'$
Msg 5. $a \to b \ : \ \{SK(a).na'.nb'\}_{PK(a)}$

The first three messages of this protocol are exactly the same as those of the fixed Needham-Schroeder Public Key Protocol. In Message 4, agent b sends a new nonce nb' to the initiator; in the final message, agent a packages up his own secret key, a new nonce of his choice, and nb', and sends all three to b encrypted under his own public key.

If the RSA weakness is not exploitable, this protocol is secure. Nothing of moment occurs in the last two messages, except for the sending of a long-term secret key; and this secret key is never sent except encrypted in such a way that this key itself is required in order to perform the decryption. (There may be type flaw attacks on this protocol; but such attacks are easily and cheaply avoided by following the implementation scheme recommended in [8].)

Even in the presence of an intruder who can exploit the RSA weakness, the protocol remains secure (see [6] for analysis of this protocol on a small network, in the presence of such an intruder). No matter how many times the intruder learns an encryption from a Message 5, he cannot find two with a known linear relation between them, because the value of na' will be different (and unknown) in each case.

However, it is not possible to find a rank function to prove that the protocol is secure in the presence of such an intruder. Consider the following protocol run, in which agent A initiates a run with the intruder, agent C:

Msg 1. $A \to C \ : \ \{A.N_A\}_{PK(C)}$
Msg 2. $C \to A \ : \ \{N_A.N_C.C\}_{PK(A)}$
Msg 3. $A \to C \ : \ \{N_C\}_{PK(C)}$
Msg 4. $C \to A \ : \ N_C{}'$
Msg 5. $A \to C \ : \ \{SK(A).N_A{}'.N_C{}'\}_{PK(A)}$

In the fourth message, the intruder chooses some nonce $N_C{}'$ to send to A, and this is reflected in the message $\{SK(A).N_A{}'.N_C{}'\}_{PK(A)}$ that C receives as a result: A chooses a nonce $N_A{}'$, and sends it to C along with C's nonce and A's secret key, all encrypted under A's public key (the inverse of which C does not hold).

Now, it is clear from the above sequence of messages that we must have

$$\rho(\{SK(A).N_A{}'.N_C{}'\}_{PK(A)}) = 1$$

for A is willing to send this message out at the end of the run: if the process representing A is to maintain the rank, then the final message that it sends out

during this protocol run must have a rank of one. But what if the intruder had chosen a different nonce, in place of $N_C{}'$? What if he had instead chosen $N_C{}''$? Then A would have sent $\{SK(A).N_A{}'.N_C{}''\}_{PK(A)}$ instead. We must therefore also have

$$\rho(\{SK(A).N_A{}'.N_C{}''\}_{PK(A)}) = 1$$

—despite the fact that, during the operation of the network, at most one of these messages can be sent out. Agent A will choose a different value for na' in each run, and will never use the same value twice; but because C can send either $N_C{}'$ or $N_C{}''$, each of the above encryptions could be (independently) available to the intruder, and so each of them must be given a rank of one. But now, of course, our new deduction rule tells us that

$$\left\{\{SK(A).N_A{}'.N_C{}'\}_{PK(A)}, \{SK(A).N_A{}'.N_C{}''\}_{PK(A)}, N_C{}', N_C{}'', PK(A)\right\} \vdash SK(A)$$

and so A's secret key must have a rank of one—as if the intruder could learn the secret key. He cannot in fact learn it, because he cannot get hold of both of these messages in order to exploit the weakness. If the intruder really could get hold of $SK(A)$, then of course he could masquerade as A as often as he pleased, and the protocol would be broken. The rank functions model cannot, it seems, be extended to deal with the attack on low-exponent RSA because its inability to distinguish mutually exclusive possibles from compossibles wreaks havoc with this type of deduction rule.

The root cause of this is the inequality test in the deduction rule. The failure to distinguish a case in which either of two (but not both) messages can be sent out from a case in which both can be sent out seems not to cause a problem when inequality tests are not permitted in the actions of the agents or the intruder; but with deduction rules such as the RSA rule, where it is advantageous to the intruder to possess two distinct messages of the same type, this failure gives the rank functions model trouble with verifying some correct protocols.

5 Strand spaces

We next turn to the strand spaces model, as described in [22, 24, 21, 23], and conduct an analogous line of enquiry: can we fit our strengthened intruder into the strand spaces model without encountering similar difficulties?

5.1 Basic definitions

A *strand* models the actions of an agent on the network, or an atomic action performed by the penetrator. (The *penetrator* in the strand spaces model corresponds to the intruder in the data independence and rank functions models.) A *regular strand* represents a run of the protocol considered from the point of view of one of the agents involved. A *penetrator strand* models an atomic action performed by the penetrator; for instance, concatenating two messages that

he knows, or sending a message out over the network. The techniques available to the penetrator in the strand spaces model are essentially the same as those available to the intruder in the rank functions and data independence models.

Definition 4. *We write 'A' to denote the space of messages that are communicable across the network. An element $t \in \mathsf{A}$ is called a* term.

Definition 5. *The set of cryptographic keys is denoted by 'K'; this set is a subset of* A.

Definition 6. *The atomic messages that are not keys form the set* T. *This set is a subset of* A. *The sets* K *and* T *are disjoint.*

Definition 7. *A* strand *is a sequence of signed terms. We write '$+t$' to represent transmission of a term t, with reception written as '$-t$'. A general strand is denoted by '$\langle \pm t_1, \ldots, \pm t_n \rangle$'.*

A strand representing an honest agent models the transmissions and receptions involving that agent in a single run of the protocol.

Definition 8. *The signed terms of a strand are called its* nodes. *The ith node of a strand s is denoted by '$\langle s, i \rangle$'.*

5.2 Strand spaces and bundles

A collection of strands may be considered as a graph, with two edge types, $\Rightarrow$ and $\rightarrow$, representing respectively consecutive terms on the same strand, and communication between two strands.

Definition 9. *If n_{i+1} immediately follows n_i on the same strand, then we write '$n_i \Rightarrow n_{i+1}$'.*

Definition 10. *If $n_1 = +a$ and $n_2 = -a$ for some term $a \in \mathsf{A}$ then we write '$n_1 \rightarrow n_2$'.*

Definition 11. *A* strand space *is then a collection of strands considered as a graph ordered by $\Rightarrow \cup \rightarrow$.*

A *bundle* in the strand space model corresponds to a trace of the whole network in the rank functions approach. It is a finite set of strands, ordered by $\Rightarrow \cup \rightarrow$, on which certain conditions are imposed to ensure that

- reception events never occur unless the corresponding transmission event has occurred;
- whenever an agent starts a protocol run, he starts from the beginning of the protocol;
- there is no backwards causation; that is, there are no loops in the graph.

Definition 12. *If $\mathcal{C} \subseteq (\rightarrow \cup \Rightarrow)$ is a finite set of edges, and $\mathcal{N}$ is the set of nodes that appear on edges in $\mathcal{C}$, then $\mathcal{C}$ will be called a* bundle *if*

- *whenever $n_2 \in \mathcal{N}$ and n_2 is a negative node, then there exists a unique $n_1 \in \mathcal{N}$ with $n_1 \rightarrow n_2 \in \mathcal{C}$;*
- *whenever we have $n_2 \in \mathcal{N}$ and $n_1 \Rightarrow n_2$, then $n_1 \Rightarrow n_2 \in \mathcal{C}$;*
- *$\mathcal{C}$ is acyclic.*

5.3 Penetrator strands

The analogue of the $\vdash$ relation in the rank functions world is a type of strand known as a *penetrator strand*. A penetrator strand represents a deduction that the penetrator may make under $\vdash$, with different types of penetrator strand corresponding to different types of deduction. In addition, since the penetrator in the strand spaces model has no local state, there will be a type of penetrator strand to represent duplicating a term in order to be able to use it twice; and since every network message that is transmitted is always received by another strand, we introduce one final type of penetrator strand to model hearing and disregarding a message.

Just as the rank functions intruder starts by knowing everything in $INIT$, so the penetrator will have some initial knowledge. However, in the strand spaces model, only the keys known to the penetrator are stated: these keys form the set K_P.

Definition 13. *A* penetrator strand *is one of the following:*

M ***Text message*** $\langle +t \rangle$ *for* $t \in \mathsf{T}$.
F ***Flushing*** $\langle -x \rangle$ *for* $x \in \mathsf{A}$.
T ***Tee*** $\langle -x, +x, +x \rangle$ *for* $x \in \mathsf{A}$.
C ***Concatenation*** $\langle -x, -y, +xy \rangle$ *for* $x, y \in \mathsf{A}$.
S ***Separation*** $\langle -xy, +x, +y \rangle$ *for* $x, y \in \mathsf{A}$.
K ***Key*** $\langle +k \rangle$ *for* $k \in K_P$.
E ***Encryption*** $\langle -k, -x, +\{x\}_k \rangle$ *for* $x \in \mathsf{A}$, $k \in \mathsf{K}$.
D ***Decryption*** $\langle -\{x\}_k, -k^{-1}, +x \rangle$ *for* $x \in \mathsf{A}$, $k \in \mathsf{K}$.

This definition is parameterised by the set T.

Recall from Definition 12 that for each negative node n_2 in a bundle there must be exactly one positive node n_1 such that $n_1 \rightarrow n_2$; that is, every reception of a message must be matched to exactly one transmission. The purpose of strands of type **F** is to allow the penetrator to 'absorb' transmissions of messages that he does not wish to use; strands of type **T** perform the corresponding rôle of replicating messages that the penetrator wants to transmit more than once.

Recent work on the strand spaces model in [4, 5] has dropped this requirement of matched transmissions and receptions, allowing communications to be sent to zero nodes, one node or many nodes. Consequently, the need for penetrator strands of types **F** and **T** has been abrogated. This slightly simplifies the notation, but does not alter the expressive power of the language. Here, we follow the notation of [21] and keep these types of penetrator strand; but this decision does not affect the analysis.

5.4 Regular strands

A *regular strand* corresponds to a run of the protocol by an honest agent, or the actions of a trusted server. It will usually be, as is the case in our model, a specific instantiation of a strand template containing free variables. The formal definition of a regular strand is very simple:

Definition 14. *A* regular strand *is any strand that is not a penetrator strand.*

5.5 Subterms and ideals

We sometimes wish to talk about the subcomponents of a compound message. The concept in the strand spaces model that allows us to do so is that of a *subterm*. The notation '$t_1 \sqsubset t_2$' will imply, informally speaking, that t_1 can be found somewhere in the makeup of t_2); but note that we shall not have $k \sqsubset \{m\}_k$ unless $k \sqsubset m$.

The $\sqsubset$ relation is defined in terms of *ideals*. Ideals in the strand spaces model allow us to talk about all messages containing a particular submessage, when encryption is restricted to a particular set of keys.

Definition 15. *Let* $\mathsf{k} \subseteq \mathsf{K}$ *be a set of keys, and* $I \subseteq \mathsf{A}$ *a set of terms. Then we say that* I *is a* k-ideal *of* A *if*

1. $hg \in I$ *and* $gh \in I$ *whenever* $h \in I$ *and* $g \in \mathsf{A}$*;*
2. $\{h\}_k \in I$ *whenever* $h \in I$ *and* $k \in \mathsf{k}$.

We write '$I_{\mathsf{k}}[h]$*' for the smallest* k*-ideal containing* h*. Similarly, if* S *is a set of terms, then '*$I_{\mathsf{k}}[S]$*' denotes the smallest* k*-ideal that includes* S*.*

Definition 16. *We say that* $h \sqsubset_{\mathsf{k}} m$ *if* $m \in I_{\mathsf{k}}[h]$*. When* $h \sqsubset_{\mathsf{K}} m$*, we drop the subscript and write simply '*$h \sqsubset m$*', and say that* h *is a* subterm *of* m*.*

5.6 Honesty

A node is an *entry point* to a set if it transmits a term without having already transmitted or received any term in the set.

Definition 17. *If, for a node* n *of a strand* s*, and a set* $I \subseteq \mathsf{A}$*,*

- n *is a positive node;*
- *the term of* n *is in* I*;*
- *no previous node on* s *has its term in* I

then we say that n *is an* entry point *to* I*.*

The idea of an *honest set* is an important one. A set is honest relative to a given bundle if the penetrator can never break into the set except by pure fluke: either he guesses the right nonce (on an **M** strand), or he guesses the right key (on a **K** strand).

Definition 18. *A set* $I \subseteq \mathsf{A}$ *is* honest *relative to a bundle* $\mathcal{C}$ *if whenever a node of a penetrator strand* s *is an entry point to* I*, then* s *is a strand of type* ***M*** *or type* ***K****.*

Although honesty is defined for sets in general, it is when the concepts of an honest set and an ideal are conjoined that they are most powerful. The main theorem from [21] gives conditions under which an ideal is honest.

Theorem 2. *Suppose*

1. $\mathcal{C}$ *is a bundle over* A*;*
2. $S \subseteq \mathsf{T} \cup \mathsf{K}$*;*
3. $\mathsf{k} \subseteq \mathsf{K}$*;*
4. $\mathsf{K} \subseteq S \cup \mathsf{k}^{-1}$*.*

Then $I_{\mathsf{k}}[S]$ *is honest.*

Proof. The proof is omitted; it may be found in [21].

5.7 Strengthening the intruder

Theorem 2 is the lynchpin of a cutting-edge strand space analysis (see [21] for examples of the theorem in action); and, in fact, strand spaces proofs prior to the introduction of this theorem were also conducted along essentially these lines. The proof of the theorem is by a case-by-case analysis of the possible types of penetrator strand, demonstrating that for each type, with the exceptions of types **M** and **K**, no strand can provide an entry point to $I_{\mathsf{k}}[S]$.

Extending the strand spaces model to cover the RSA attack will involve introducing a new type of penetrator strand

L ***Low-exp RSA*** $\langle -\{a.X.b\}_k, -\{c.X.d\}_k,$
$-k, -a, -b, -c, -d, +X\rangle$ for $a, b, c, d, X \in \mathsf{A}$; $a \neq c$ or $b \neq d$; and $k \in \mathsf{K}$

modelling the penetrator's new-found ability to exploit the weakness. Before we can use this extended model to prove correctness of security protocols even in the presence of our new penetrator, we are required to show that the standard strand spaces results still hold good. In particular, this means that we need to extend the proof of Theorem 2 to include penetrator strands of type **L**.

This, sadly, cannot be done. Let $\mathcal{C}$ be a bundle over A, and let $S = \{X, K^{-1}\}$, with $K \in \mathsf{K}$, $\mathsf{k} = \mathsf{K} \setminus \{K\}$ and $X \in \mathsf{T}$. Now let $a, b, c, d \in \mathsf{T} \setminus S$. The conditions of Theorem 2 are now satisfied, and so we should have that $I_{\mathsf{k}}[S]$ is honest. However, suppose that $\mathcal{C}$ contains the strand

$$\langle -\{a.X.b\}_K, -\{c.X.d\}_K, -K, -a, -b, -c, -d, +X\rangle$$

representing the penetrator's deduction of X by making use of the RSA attack. Now the first two terms lie outside the ideal, because $K \notin \mathsf{k}$; the next five are not in the ideal because they are not contained in S. But the last term, $+X$, is positive, and $X \in I_{\mathsf{k}}[S]$. So this strand provides an entry point for $I_{\mathsf{k}}[S]$, contradicting the definition of honesty.

It appears, then, that there is no simple way to extend the strand spaces model to cover this RSA attack without falsifying the central result used to verify protocols. It is, again, the implicit inequality test that lies at the root. For the concept of an honest ideal captures the notion of the set of messages such that it is undesirable to allow the penetrator to learn any of the messages in the set: if the penetrator can pick up a single message from inside $I_{\mathsf{k}}[S]$, for

some suitable set k, he will be able to learn the values of some members of S. Allowing for the RSA weakness would require specifying a set H such that the penetrator should be able to pick up *no more than one* message from H—or, in other words, if he can learn two messages c_1 and c_2 from H, *and* we have that $c_1 \neq c_2$, then he can make use of the attack. Incorporating this into the strand spaces model would require substantial reworking of the basic tools of the trade; and, in any case, it is not at all clear how it could be done.

6 Conclusion

The data independence model cannot be easily adapted to allow the intruder to take advantage of the weakness in low-exponent RSA. We have shown in this paper that both the rank functions method for CSP models, and the use of honest ideals in the strand spaces model, suffer from the same inflexibility, and that it is the implicit inequality checks that are the cause in each case.

The fact that all three models fail to allow for this attack, and all for the same reason, suggests that the mechanisms by which these three approaches to security protocol analysis manage to deal with unbounded networks have more similarities than meet the eye.

Each approach involves identifying an infinite set of messages that can all be treated in the same way. The data independence model requires a data type T such that replacing one member of T for another throughout the system should not defeat any attacks. The rank functions model identifies sets of messages such that the intruder gains no more information from learning all members of the set than he does from learning just one, and so simply keeps track of whether the intruder knows the whole set or none of it. The strand spaces model operates in much the same way: an honest ideal $I_{\mathsf{k}}[S]$ is the set of messages any of which can be used by the penetrator to learn something from the set S; and so the requirement is always to show that none of the messages in the ideal can be picked up by the penetrator. What none of the models can allow for is the idea that allowing the intruder to learn *one* of the messages might be acceptable, whereas to reveal *two* (or more) will result in a breach of security—or, more appropriately, that it is dangerous to allow the intruder to learn two such messages c_1 and c_2 if and only if $c_1 \neq c_2$. We simply cannot deal with the inequality test.

We suspect that Roscoe and Broadfoot's notion of a positive deductive system will accurately limit what can be analysed not just with the data independence model, but with each of the three models considered in this paper. Formalising this idea within the rank functions and strand spaces models, and proving that positive deductive systems do indeed define the scope of what can be verified using these models, is the subject of ongoing research.

We are interested to note that Stoller's approach [20] to identifying bounds on the number of protocol runs required for an attack can be extended to deal with the inequality tests required for the attack on RSA considered here, under certain mild conditions on the nature of the protocol under consideration. This implies

that Stoller's method is different at a deeper level from the three approaches considered in this paper, and gives hope that alternative techniques for CSP models and strand spaces might be developed that can handle inequality tests.

Acknowledgements Thanks are due to Bill Roscoe for his helpful comments on a draft of this paper.

References

1. D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter. Low-exponent RSA with related messages. *Lecture Notes in Computer Science*, 1070, 1996.
2. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.
3. M. Franklin and M. Reiter. A linear protocol failure for RSA with exponent three. 1995. Presented at the Rump Session of Crypto '95, Santa Barbara, CA.
4. Joshua D. Guttman and F. Javier Thayer Fábrega. Authentication Tests. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*. IEEE Computer Security Press, May 2000.
5. Joshua D. Guttman and F. Javier Thayer Fábrega. Protocol Independence through Disjoint Encryption. *Proceedings of 13th IEEE Computer Security Foundations Workshop*, pages 24–34, June 2000.
6. James A. Heather. Exploiting a weakness of RSA. Master's thesis, Oxford University Computing Laboratory, September 1997.
7. James A. Heather. *'Oh! ... Is it really you?'—Using rank functions to verify authentication protocols.* Department of Computer Science, Royal Holloway, University of London, December 2000.
8. James A. Heather, Gavin Lowe, and Steve A. Schneider. How to avoid type flaw attacks on security protocols. *Proceedings of 13th IEEE Computer Security Foundations Workshop*, June 2000.
9. James A. Heather and Steve A. Schneider. Towards automatic verification of authentication protocols on an unbounded network. *Proceedings of 13th IEEE Computer Security Foundations Workshop*, June 2000.
10. C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
11. Ranko S. Lazić. *A semantic study of data-independence with applications to the mechanical verification of concurrent systems.* PhD thesis, University of Oxford, 1997.
12. Ranko S. Lazić. Theorems for Mechanical Verification of Data-Independent CSP. Technical report, Oxford University Computing Laboratory, 1997.
13. Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. In *Proceedings of TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.
14. Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.
15. Ron L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

16. A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall International, 1998.
17. A. W. Roscoe and Philippa J. Broadfoot. Proving security protocols with model checkers by data independence techniques. *Journal of Computer Security*, 1999.
18. Steve A. Schneider. Verifying authentication protocols in CSP. *IEEE TSE*, 24(9), September 1998.
19. Steve A. Schneider. *Concurrent and real-time systems: the CSP approach.* John Wiley & Sons, 1999.
20. Scott D. Stoller. A bound on attacks on authentication protocols. In *Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science (TCS)*, Kluwer, 2002.
21. F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Honest ideals on strand spaces. *Proceedings of 11th IEEE Computer Security Foundations Workshop*, June 1998.
22. F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, May 1998.
23. F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Mixed strand spaces. *Proceedings of 12th IEEE Computer Security Foundations Workshop*, pages 72–82, June 1999.
24. F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2,3):191–230, 1999.