On the Security of Ping-Pong Protocols when Implemented using the RSA

(Extended Abstract)

Shimon Even [1] Oded Goldreich [1,2] Adi Shamir [3]

ABSTRACT

The Security of the RSA implementation of ping-pong protocols is considered. It is shown that the obvious RSA properties, such as "multiplicativity", do not endanger the security of ping-pong protocols. Namely, if a ping-pong protocol is secure in general then its implementation using an "ideal RSA" is also secure.

1. INTRODUCTION

When studying the security of cryptographic protocols, one can take one of the following two approaches:

1) Distinguish between the security of the "high level structure" of the protocol and the security of the cryptosystems used for its implementation. The aim is, mainly, to better understand the structure of secure protocols and issues related to it. While studying the (security of the) structure of a protocol, it is assumed that the protocol is "implemented" with "ideal" cryptosystems. In other words, the cryptosystems are treated as if they were free of any properties which are not implied by the cancellation of encryption with the corresponding decryption. Such a treatment has usually an algebraic flavour.

   This approach can be found in [NS], [DY], [DLM], [DEK], [EG] and [EGL].

2) Study the security of a concrete implementation of the protocol with respect to the concrete cryptosystems used for the implementation. The aim is to develop concrete provably-secure protocols and to present a methodology for

[1] *Computer Science Dept., Technion, Haifa, Israel.*

[2] Currently in MIT, Lab. for Computer Science. Supported in part by a Weizmann Postdoctoral Fellowship.

[3] Department of Applied Mathematics, Weizmann Institute, Rehovot, Israel.

developing and proving correctness of protocols. Characteristic tools in this approach are generalized notions of polynomial-time reductions.

This approach was pursued in [LMR], [GMR], [BGMR], [ACGM], [CF] and [GHY].

In this paper, we follow the first approach, but introduce some influences of the second approach. More specifically, we study the "high level structure" of protocols implemented using "ideal-RSA" cryptosystems (i.e. cryptosystems which posses only the obvious properties of the RSA). Our aim is to try to characterize the structure of protocols which are secure with respect to the obvious properties of the RSA. We restrict our study to a simple class of public-key protocols, known as ping-pong protocols. The reason for this restriction is that testing the security of protocols, from a slightly extended class, has been shown to be undecidable [EG]. We show that as far as the security of ping-pong protocols is concerned the obvious properties of the RSA do not give an adversary any additional edge. Put in other words, ping-pong protocols which are secure with respect to "ideal cryptosystems" - remain secure with respect to "ideal-RSA".

Our work was partially motivated by Denning's study of the weaknesses of the RSA implementation of a simple signing protocol [D]. We show that the weaknesses, pointed out in [Da] and [D], are due to the insecurity of the "high level structure" of the protocol and not to the fact that it was implemented using the RSA. We further discuss this issue in section 7.

2. PING-PONG PROTOCOLS AND THEIR SECURITY

In this section we recall the basic definitions regarding ping-pong protocols and their security problem.

2.1 Public Key Cryptosystems

Following [DH], a *public key cryptosystem (PKCS)* is a set of pairs of functions, such that every user X has an encryption function E_X and a decryption function D_X. Both functions are mappings from $\{0,1\}^*$ to $\{0,1\}^*$. There is a public directory containing all (X, E_X) pairs, while the decryption function D_X is known only to X. It is required that

(1) For every $m \in \{0,1\}^*$, $E_X(D_X(m)) = D_X(E_X(m)) = m$.
D_X is the inverse function of E_X.

(2) It is infeasible to recover z when given $E_X(z)$ (and E_X).
For further details consult [DH] and [RSA].

In the rest of this paper we will refer to the encryption and decryption function as to operators. Operator words are defined (as usual) as the composition of operators; i.e. the operator word $\sigma_t \cdots \sigma_2 \sigma_1$ maps $m \in \{0,1\}^*$ to $\sigma_t(\cdots \sigma_2(\sigma_1(m))\cdots)$. Two operator words α and β are said to be equivalent if for every $m \in \{0,1\}^*$, $\alpha(m)=\beta(m)$. The equivalence between operator words will be denote by $\equiv$.
Property (1) above implies the following

Operators' Cancellation Rules:
for every X, $E_X D_X \equiv D_X E_X \equiv$ the identity operator.

Property (2) above implies that user X can only apply encryption operators and his own decryption operator (i.e. operators from the set $\{D_X\} \bigcup \{E_Y$: Y is any user in the network$\}$.)

2.2 Ping-Pong Protocols

Following [DY], a *ping-pong protocol* $P(X,Y)$ is a sequence $(\alpha_1,\alpha_2, \ldots, \alpha_l)$ of operator words, such that $\alpha_{2i-1} \in \{D_X, E_X, E_Y\}$ and $\alpha_{2i} \in \{D_Y, E_X, E_Y\}$. Here X and Y are variables. In a concrete execution of the protocol they are substituted by the names of the participants.

An execution of protocol $P(\cdot,\cdot)$ by parties A and B, regarding the *initial message* $m_0 \in \{0,1\}$, proceeds as follows: In the first phase party A applies $\alpha_1[A,B]$ to the initial message m_0 , and transmits the result to B. In other words, in the first phase A transmits $m_1 = \alpha_1[A,B](m_0)$ to B. In the $2i$-th phase B applies $\alpha_{2i}[A,B]$ to m_{2i-1} , and transmits the result ($m_{2i} = \alpha_{2i}[A,B](m_{2i-1})$) to A. In the $2i+1$-st phase A applies $\alpha_{2i+1}[A,B]$ to m_{2i} , and transmits the result ($m_{2i+1} = \alpha_{2i+1}[A,B](m_{2i})$) to B. Here $\alpha_i[A,B]$ denotes the operator word which results from α_i by substituting E_X $[D_X]$ by E_A $[D_A]$ and E_Y $[D_Y]$ by E_B $[D_B]$.

2.3 Security of Ping-Pong Protocols

Following [DY], we say that a ping-pong protocol $P(\cdot,\cdot)$ is insecure if parties which did not take part in an execution of P (hereafter referred to as the *saboteurs*) can find out the initial message transmitted in that execution. To this end the saboteurs can initiate other executions of P and rely on the operators' cancellation rules (i.e. $E_X D_X \equiv D_X E_X \equiv$ the identity operator). It was shown [DEK] that it is sufficient to consider a single saboteur. A formal definition of insecurity follows:

Definition 1: Let $P(X,Y) = (\alpha_1,\alpha_2, \ldots, \alpha_l)$ be a ping-pong protocol (as in the previous subsection).
Let A, B and S denote three distinct users.
Let $\Sigma_Z = \{D_Z\} \bigcup \{E_U: U$ is any user name $\}$. (Σ_X denotes the set of operators which may be applied by user X.)
Let $I(A,B,S) = \{\alpha_i[X,Y]: 1 \leq i \leq l$ and $X \neq Y \in \{A,B,S\}\}$. ($I(A,B,S)$ is the set of operator words which may be effected on messages in executions of the protocol $P(\cdot,\cdot)$ by two users out of A, B and S.)
Let $\Delta = (\Sigma_S \bigcup I(A,B,S))^*$.
The protocol $P(\cdot,\cdot)$ is *insecure* if there exists an operator word $\gamma \in \Delta$ such that $\gamma\alpha_1[A,B]$ is equivalent (under the operators' cancellation rules) to the identity operator. The operator word $\gamma\alpha_1[A,B]$ is called an *insecurity string.*

Remark 1: The above definition (in which only one saboteur is considered) is equivalent to a definition in which more than one saboteur is considered. The latter definition can be obtained by redefining Δ as follows

$$\Delta = ((\bigcup_{Z \neq A,B} \Sigma_Z) \bigcup (\bigcup_{X \neq Y \neq Z \neq X} I(X,Y,Z)))^*$$

A proof of the equivalence of the two definitions can be found in [DEK]. It is interesting to note that this equivalence does not hold for ping-pong protocols for more than two parties [EG].

Discussion: Note that under (the insecurity) Definition 1, the only properties of the public-key cryptosystem exploited by the saboteur (in his attack on the protocol) are the most obvious and general ones; namely the cancellation of encryption with the corresponding decryption. Definition 1 can be interpreted as considering only the security of the "high level structure" of the protocol.
Testing "high level security", may obviously provide evidence for the insecurity of a concrete implementation of the protocol, but it can not provide a proof that a particular implementation (with a particular public-key cryptosystem) is secure.

3. THE RSA AND ITS PROPERTIES

The RSA is the most popular implementation of the concept of a public-key cryptosystem. This system, presented in 1978 by Rivest, Shamir and Adleman [RSA] is widely believed to be secure. However, the encryption decryption functions of the RSA possess obvious properties which are not implied by the cancellation rules. We begin by presenting the RSA functions and continue by discussing their properties.

3.1 The RSA Functions

An instance of the RSA consists of a composite integer N which is the product of two large primes p and q, and two integers e and d such that $e \cdot d$ is congruent to 1 modulo $\phi(N)$ ($\phi(N) = (p-1)(q-1)$ is the Euler function).

To create an instance of the RSA, user A randomly picks two large primes p and q, and a number relatively prime to $(p-1)(q-1)$. User A computes $N = p \cdot q$ and $d = e^{-1} \; mod \; (p-1)(q-1)$. User A places $(A \;, (N, e))$ in the public directory and keeps all other information (in particular d, p and q) secret.

Encryption is done by raising the message to the e-th power modulo N; while decryption is done by raising the message to the d-th power modulo N. Everyone can encrypt a message so that A can decrypt it. It is assumed that knowledge of the factorization of N is needed in order to be able to decrypt (and factorization is considered intractable). For simplicity let us identify the username (i.e. A) with the modulos N he uses. The encryption function of user N will be denoted by E_N and N's decryption function will be denoted by D_N.

Formal Setting: Let us denote by Z_N the set of all residues modulo N (i.e. $Z_N = \{0,1,2,...,N-1\}$). By the above we have, for every $m \in Z_N$,

$$E_N(m) = m^e \; mod \; N \quad \text{and} \quad D_N(m) = m^d \; mod \; N$$

It can be easily shown that E_N and D_N cancel each other [RSA].

In addition to the cancellation of encryption by the corresponding decryption, the RSA possesses additional obvious relations - to be hereby discussed.

3.2 The Properties of the RSA

The main properties of the RSA are that the set of almost all its message space forms a group with respect to multiplication modulo N, and that the encryption and decryption operators are homomorphisms over this group. Note that the RSA induces a permutation over Z_N^* . For simplicity, we restrict the message space to $Z_N^* \subseteq Z_N$. This excludes only $p+q-1$ elements - a negligible fraction of the original message space (Z_N).

In subsection 3.3 it will be shown that all the other obvious properties of RSA can be derived from the above. This includes the fact that D_N is a homomorphism, the fact that $E_N(1)=1$ etc.

3.3 Axiomatization

In this subsection we present a complete axiomatization of the RSA properties discussed above. In the formal treatment, we will denote the message space by M_X . Recall that E_X and D_X are inverse permutations over M_X . A multiplication operator over M_X will be considered. It is axiomatized that this operator (denoted by μ_X) together with the set M_X forms an Abelian group. It is also axiomatized that E_X is a homomorphism of this group.

A0) *Cancellation Axiom*: For every $m \in M_X$ the following holds

$$D_X(E_X(m)) = E_X(D_X(m)) = m\,.$$

A1) *Abelian Group Axiom*: The set M_X and the binary operation μ_X form an Abelian group. That is, $\mu_X : M_X \times M_X \to M_X$ satisfies the followings (for every $m, m_1, m_2, m_3 \in M_X$):

A1.1) $\mu_X(m_1, \mu_X(m_2, m_3)) = \mu_X(\mu_X(m_1, m_2), m_3)$

A1.2) $\mu_X(1,m) = \mu_X(m,1) = m\,.$

A1.3) There exists a $m^{-1} \in M_X$, $\mu_X(m, m^{-1}) = \mu_X(m^{-1}, m) = 1$

A1.4) $\mu_X(m_1, m_2) = \mu_X(m_2, m_1)$

A2) *Homomorphism of the Encryption*: For every $m_1, m_2 \in M_X$ the following holds

$$E_X(\mu_X(m_1, m_2)) = \mu_X(E_X(m_1), E_X(m_2))$$

An equivalent formulation is achieved by generalizing the multiplication operator μ_X, to take arbitrary many arguments, and by introducing the multiplicative inverse function I_X.

The RSA Equalities

E0) *Cancellation of Encryption/Decryption*: For every $m \in M_X$, $D_X(E_X(m)) = m$ and $E_X(D_X(m)) = m$.

E1) *Nested Multiplication*: For every $m_1, m_2, \ldots, m_d \in M_X$, and $0 < j-i < d$

$\mu_X(m_1, \ldots, m_i, \mu_X(m_{i+1}, \ldots, m_j), m_{j+1}, \ldots, m_d) = \mu_X(m_1, m_2, \ldots, m_d)$.

E2) *Redundant Multiplication*: For every $m \in M_X$, $\mu_X(m) = m$.

E3) *Redundant Identity*: For every $m_1, \ldots, m_d \in M_X$, and $d \geq 1$

$\mu_X(m_1, \ldots, m_i, 1, m_{i+1}, \ldots, m_d) = \mu_X(m_1, \ldots, m_i, m_{i+1}, \ldots, m_d)$

E4) *Inverse*: For every $m, m_1, \ldots, m_d \in M_X$, $d \geq 0$ and $i \leq j$,

$$\begin{aligned}&\mu_X(m_1, \ldots, m_i, m, m_{i+1}, \ldots, m_j, I_X(m), m_{j+1}, \ldots, m_d) = \\ &\mu_X(m_1, \ldots, m_i, I_X(m), m_{i+1}, \ldots, m_j, m, m_{j+1}, \ldots, m_d) = \\ &\qquad \mu_X(1, m_1, \ldots, m_i, m_{i+1}, \ldots, m_j, m_{j+1}, \ldots, m_d)\end{aligned}$$

E5) *Homomorphism of Inverse Operator*: For every $m_1, m_2, \ldots, m_d \in M_X$, and $d \geq 2$, $I_X(\mu_X(m_1, m_2, \ldots, m_d)) = \mu_X(I_X(m_1), I_X(m_2), \ldots, I_X(m_d))$.

E6) *Cancellation of Double Inverse*: For every $m \in M_X$, $I_X(I_X(m)) = m$.

E7) *Generalized Homomorphism of Encryption/Decryption*:
For every $m_1, m_2, \ldots, m_d \in M_X$ and $d \geq 2$ the following hold

$$E_X(\mu_X(m_1, m_2, \ldots, m_d)) = \mu_X(E_X(m_1), E_X(m_2), \ldots, E_X(m_d))$$
$$D_X(\mu_X(m_1, m_2, \ldots, m_d)) = \mu_X(D_X(m_1), D_X(m_2), \ldots, D_X(m_d))$$

E8) *Stability of Identity*: $E_X(1)=1$ $D_X(1)=1$ and $I_X(1)=1$.

E9) *Commutativity of Operators*: For every $m \in M_X$,
$E_X(I_X(m))=I_X(E_X(m))$ and $D_X(I_X(m))=I_X(D_X(m))$.

4. SECURITY WITH RESPECT TO RSA PROPERTIES

In this section we define a new notion of insecurity: insecurity w.r.t RSA. Loosely speaking, a protocol is insecure w.r.t RSA if an adversary can seize the initial message by eavesdropping, initiating other executions of the protocol and taking advantage over the (10) equalities listed above.

In order to formally discuss the power of such an adversary, we have to study the algebra of expressions over the operator alphabet $\bigcup_X \{E_X, D_X, \mu_X, I_X\}$ w.r.t the equalities listed in Sec. 4.2. This algebra is best described by representing its expressions as rooted labelled trees and enforcing its equalities by tree manipulation rules.

4.1 The Algebra of Operator Trees

We start the description of the algebra by giving a representation of its expressions as rooted node-labelled trees.

Definition 2: An *operator tree* is recursively defined as follows:

A *constant* is an element of $\bigcup_X M_X$.

A *variable* may be assigned any element of $\bigcup_X M_X$.

An *atom* is a node labelled either a constant or a variable. An atom is an operator tree (rooted at the atom).

A *protocol node (P-node)* is a node labelled either E_X or D_X for some X. An operator tree rooted at a P-node v consists of the node v, an edge (v, u) and an operator tree rooted at u. The operator tree rooted at u is said to be a subtree hooked to v.

An *inverse-node (I-node)* is a node labelled I_X for some X. An operator tree rooted at an I-node v consists of the node v, an edge (v,u) and an operator tree rooted at u. (The operator tree rooted at u is said to be a subtree hooked to v.)

An *multiplication-node* (μ-node) is a node labelled μ_X for some X. An operator tree rooted at an μ-node v consists of the node v, a set of $d \geq 1$ edges $\{(v,u_i)\}_{i=1}^{d}$ and a set of operator trees rooted at u_1 , u_2 ... u_d respectively. (The operator tree rooted at u_i is said to be a subtree hooked to v. Note that only a μ-node may have more than one son in an operator tree.)

As a first step towards defining the operator tree algebra we define two operator trees to be isomorphic if there is a "labelling and rooting preserving" isomorphism from one tree to the other. This isomorphism can be precisely defined as follows:

Definition 3: Two operator trees T_1 and T_2 are said to be *isomorphic* if one of the following hold:

1) Both trees are atoms, and either both are labelled by the same constant or both are labelled by the same variable.
2) For $i \in \{1,2\}$, let T_i consist of the root v_i and d subtrees hooked to v_i denoted by t_i^1 , t_i^2 ,..., t_i^d respectively.
   Then the labelling of v_1 and v_2 are equal and there exists a permutation π (over the set $\{1,2,...,d\}$) such that for every $1 \leq j \leq d$, the subtree t_1^j is isomorphic to the subtree $t_2^{\pi(j)}$.

The equalities listed in Sec. 3.3 imply the following tree manipulation system. The system consists of 10 pairs of reduction rules, corresponding to these 10 equalities.

The Two-Way Reduction Rules: The notation $e_1(t) \longrightarrow e_2(t)$ $[e_1(t) <\!\!-\, e_2(t)]$ means that the subtree described by the expression $e_1(t)$ $[e_2(t)]$ can be replaced by the subtree described by $e_2(t)$ $[e_1(t)]$, in any operator tree. For every equality E*i* $(0 \leq i \leq 9)$ of the form $e_1(t)=e_2(t)$, we introduce a reduction rule (denoted R*i*) $e_1(t) \longrightarrow e_2(t)$ and a (reverse) reduction rule (denoted B*i*) $e_1(t) <\!\!-\, e_2(t)$.

Finally, we define equivalence of operator trees as follows

Definition 4: Two operator trees are *(two-way) equivalent* if applying a sequence of reduction rules to one of them results in an operator tree which is isomorphic to the other. We stress that both the B*i* and the R*i* reduction rules may be used in this sequence of applications.

4.2 Properties of the R-Reductions Rules

The reduction rules discussed in subsection 4.1 consist of pairs Ri and Bi such that if Ri is $e_1(t) \longrightarrow e_2(t)$ then Bi is $e_1(t) <\!\!-\; e_2(t)$. This system of reduction rules is clearly infinite, in the sense that one can apply an infinite sequence of reduction rules to every operator tree. We will consider the Ri $(0 \leq i \leq 9)$ reduction rules hereafter called the *R-reduction rules.* The system of R-reduction rules is finite; that is, for every operator tree t there is a (finite) upper bound of the length of sequences of R-reduction rules which can be applied to t.

Lemma 3: Let n denote the number of nodes in the operator tree t. Then n^3 is an upper bound on the length of R-reduction sequences which can be applied to t.

Remark 2: The upper bound presented in Lemma 3 is tight up to a multiplicative constant. A demonstration of this fact is omitted from this extended abstract.

The finiteness of the R-reduction rules suggests the following

Definition 5: An operator tree is said to be *irreducible* if no R-reduction rule can be applied to it.

Corollary 1 (to Lemma 3):

For every operator tree t, there exists an operator tree r such that

1) r is an irreducible operator tree.

2) r is the result of applying a finite sequence of R-reduction rules to t.

Another appealing feature of the R-reduction rules is the insignificance of the order in which R-reduction rules are applied;

Lemma 4:

For every operator tree t there exists a **unique** operator tree r such that

1) r is an irreducible operator tree

2) r is the result of applying a finite sequence of R-reduction rules to t.

Lemma 4 can be proven by demonstrating[4] that the R-reduction rules have the Church-Rosser Property [CR], and by using the Church-Rosser Theorem. By the Church-Rosser Theorem if a reduction system is finite (in the sense of Lemma 3) and has the Church-Rosser Property then each object has a unique irreducible object reachable from it. Lemma 4 suggests the following

Definition 7: The *reduced form* of t is an operator tree r such that r is irreducible and $t \longrightarrow {}^{*} r$.

[4] Such a demonstration is straightforward but tedious. An extensive study of general methods for proving Church-Rosser property of tree manipulation systems was conducted by Rosen [R].

Definition 8: Two operator trees are said to be *R-equivalent* if their corresponding reduced forms are isomorphic.

4.3 R-Reduction Rules versus Two-Way Reduction Rules

In this subsection we show that the R-reduction rules are as powerful as the two-way reduction rules. This is of much importance since, unlike the two-way reduction rules, the R-reduction rules can only be applied a finite number of times and yield a unique (irreducible) result.

Lemma 5: Two operator trees are R-equivalent if and only if they are two-way equivalent.

Corollary 4 (to Lemma 5): With respect to any operator trees t_1 and t_2 , the following are equivalent:

1) t_1 and t_2 can be proven to be equal in the proof system which consists of the axioms A0, A1 and A2 (of subsection 3.3) and the proof rule known as substitution.

2) The reduced form of t_1 is isomorphic to the reduced form of t_2.

Thus, the R-equivalence "grasps" the structure of the algebra of operator trees.

4.4 The New Security Definition

Having presented the algebra of operator trees and its properties, we are ready to define insecurity with respect to this algebra. We will say that a protocol is insecure if a saboteur can construct an operator tree which is R-equivalent to the initial message sent from A to B. As in Definition 1 (subsection 2.3), the saboteur can apply any operator in his vocabulary as well as apply any instance of any protocol word to any message. Let us present a formal definition of this new insecurity notion.

Definition 9: Let $P(X,Y) = (\alpha_1,\alpha_2, \ldots, \alpha_t)$ be a ping-pong protocol (as in subsection 2.2).
The protocol $P(\cdot,\cdot)$ is *RSA-insecure* if a saboteur S who does not know the initial message m_0 (in the execution of P by A and B) can construct an operator tree which is R-equivalent to m_0. (This operator tree will be called the *insecurity tree* of protocol P.)
The *trees that* S can construct are recursively defined as follows:

1) S can construct the path $\alpha_1[A,B](m_0)$

2) S can construct an atom labelled by a constant.

3) Let t be an operator tree which can be constructed by S. Then S can construct the operator tree $D_Y(t)$ ($Y \neq A, B$ is any user in the net other than A or B) and the operator tree $E_X(t)$ (X is any user in the net).

4) Let t be an operator tree which can be constructed by S. Then S can construct the operator tree $\alpha_i[X,Y](t)$, where $1 \leq i \leq l$ and $X \neq Y$ are any two distinct users.

5) Let $t_1, t_2, \ldots, t_d$ be $d \geq 2$ operator trees which can be constructed by S. Then S can construct the operator tree $\mu_X(t_1, t_2, \ldots, t_d)$ and the operator tree $I_X(t_1)$ (X is any user in the net).

Consider the following variant of the above definition:

Definition 10: The protocol $P(\cdot,\cdot)$ is *generically-insecure* if a saboteur S who does not know the initial message m_0 (in the execution of P by A and B) can construct an operator tree which is R-equivalent to m_0, where the trees that S can construct are recursively defined by 1, 2 ,3 and 4 above (without 5).

Remark 3: Note that the Definition 10 is identical to Definition 1 (the insecurity definition which appears in subsection 2.3).
(Note that if P is generically insecure then it has an insecurity tree which consists of a path with one atom (m_0) and all other nodes are P-nodes. Also recall Remark 1 in subsection 2.3.)

5. EQUIVALENCE OF THE TWO SECURITY DEFINITIONS

We are now ready to present the main result of this paper: the equivalence of the insecurity definitions presented in Definition 1 and Definition 9 (subsections 2.3 and 4.4) respectively. By Remark 3 (above) it suffices to show that

Main Theorem: A protocol P is RSA-insecure if and only if it is generically-insecure.

proof: Clearly, if the protocol P is generically-insecure then it is RSA-insecure. It is left to show that if P is RSA-insecure then it is generically-insecure.

Suppose that P is RSA-insecure, then it has an insecurity tree t which contains atoms, P-nodes and possibly I-nodes and μ-nodes. Consider an application of a R-reduction sequence to the operator tree t, resulting in the operator tree m_0. Consider the node in t that was not reduced during this reduction process (it is labelled by m_0); and the path in t from the root to this node. Denote the nodes on this path by $n_0, n_1, n_2, \ldots,$ and n_l . n_0 denotes t's root and n_l the node labelled by m_0 which was not reduced in the reduction sequence.

First note that the path which results from $n_0, n_1, \ldots, n_l$ by omitting all μ-nodes and I-nodes can be constructed by the saboteur S. It is left to show that this path (the path which results from $n_0, n_1, \ldots, n_l$ by omitting all μ-nodes and I-nodes) is R-equivalent to m_0. That is, that the P-nodes of this path can be paired in a non-interlacing manner such that the labels of the nodes of each pair are E_X and D_X (for some user X).

Throughout the R-reduction process, consider the path between the current root and n_l . Before the first reduction rule was applied this path consists of $n_0, n_1, \ldots, n_l$. Consider the application of the ith R-reduction rule.

Case I: If the ith rule is not R0 then it does not cause the omission (or insertion) of any P-node in the path from the current root to n_l . Furthermore, it also does not change the order in which the P-nodes appear on the path from the current node to n_l .
[Note that we rely on the fact that n_l was not omitted during the R-reduction sequence.]

Case II: If the ith rule is R0, but it is not applied to a node on the path from the current root to n_l , then it does not effect the nodes on this path.

Case III: If the ith rule is R0 and it is applied to n_j which is on the path between the current root and n_l , then it causes the omission of n_j and some n_k . Note that both nodes are currently adjacent on the path between root and n_l . In this case we *pair* the node n_j with the node n_k .

Note that in the end of the reduction process the path between the current root and n_l consists of a single node: n_l . It is evident that we have paired all the P-nodes of the initial path between the root and n_l so that the pairs do not interlace and the labels of the P-nodes of each pair are E_X and D_X for some X.

We conclude by noting that concatenating the labels of the P-nodes on the initial path from root to n_l forms an insecurity string of the protocol P. Thus, P is generically-insecure.

QED

6. EXTENSIONS

The definitions and results of the previous sections can be extended in three directions:

1) The security of ping-pong protocols over extended operator-alphabet.
2) The security of multi-party ping-pong protocols.
3) Insecurity as the ability to accomplish the effect of a specified operator word.

In this extended abstract, we only deal with the last case. The first two cases will appear in the full version of this paper.

Insecurity as the Ability to Apply a Specific Operator Word

The insecurity definitions appearing in this paper can be rephrased as follows: Can a saboteur construct an operator tree which is equivalent, modulo a specific set of reduction rules, to the left inverse of the first word in the protocol (i.e. equivalent to $\alpha_1[A,B]^{-1}$) ? [The reader is referred to subsection 4.4 for a definition of the set of operator trees which can be constructed by the saboteur.]

Following [E], we generalize the above notions of security, and consider the following question: Can a saboteur construct an operator tree which is equivalent, modulo a specific set of reduction rules, to a specific operator word β?
In case the saboteur can construct (using instances of the words of the protocol) such an operator word, we will say that the protocol is β-insecure, otherwise we say that the protocol is β-secure.

Using the appropriate sets of reduction rules, we derive definitions for β-RSA-insecurity and β-generic-insecurity. (In the first the set of reduction rules consists of all R reductions, while in the second the set of reduction rules consists only of R0.) The proof of the Main Theorem can be modified to yield the following:

Theorem 2: For every β, a protocol is β-RSA-insecure iff it is β-generically-insecure.

7. CONCLUDING REMARKS

In [D], Denning considered a signing protocol which consists of applying the user's decryption operator to the given document. Formally, this protocol consists of two phases: in the first one party S sends his counterparts (R) a document m (to be signed); in the second phase R applies D_R to m and replies with the result $D_R(m)$, which is considered to be R's signature to m.

Using the fact that the message space of the RSA forms a group and that the encryption function is a homomorphism of this group, Denning demonstrated (by methods of Davida [Da] and others) that the RSA implementation of the above protocol is insecure. Namely, that S can get R's signature to message m without R being willing to sign m.

The fact that the above protocol is D_R-RSA-insecure should be no surprise, since this protocol is obviously D_R-generically-insecure. In the above protocol, R does not have the choice of which messages he signs since he is assumed to play the protocol for any initial message. In order to get R's signature to m all S needs to do is to send m to R in the first phase. Thus, the weaknesses pointed out by Denning *reflects* only *the weakness of the protocol* she *used, not a weakness of the RSA*. Furthermore,

> Any successful attack on a ping-pong protocol which relies on the obvious properties of the RSA (listed in Sec. 2) can be transformed into a successful attack which works no matter which public-key cryptosystem is used to implement the protocol.
>
> In other words, *generically-secure ping-pong protocols are immune against attacks which rely on the obvious properties of the RSA.*

ACKNOWLEDGEMENT

We wish to thank Silvio Micali for many intriguing discussions concerning the security of cryptographic protocols.

REFERENCES

[ACGM]
Awerbuch B., Chor B., Goldwasser S., and Micali S., "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science,*

[BGMR]
Ben-Or, M., Goldreich, O., Micali, S., and Rivest, R.L., "A Fair Protocol for Signing Contracts", *Proc. of the 12th ICALP,* Lecture Note in Computer Science (194) Springer Verlag, 1985, pp. 43-52.

[CR] Church, A., and Rosser, J.B., "Some Properties of Conversion", *Trans. Amer. Math. Soc. 39,* (1936), pp. 472-482.

[CF] Cohen J.D., and Fischer, M.J., "A Robust and Verifiable Cryptographically Secure Election Scheme", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science,*

[Da] Davida G.I., "Chosen Signature Cryptoanalysis of the RSA (MIT) Public Key Cryptosystem", Tech. Rep. TR-CS-82-2, Dept. of Electrical Engineering and Computer Science, Univ. of Wisconsin, Milwaukee, WI, Oct. 1982.

[D] Denning D.E., "Digital Signatures with RSA and Other Public-Key Cryptosystems", *Comm. of the ACM*, Vol. 27, April 1984, pp. 388-392.

[DLM] DeMillo, R., Lynch, N., and Merritt, M., "Cryptographic Protocols", *Proc. of the 14th ACM Symp. on Theory of Computation*, 1982, pp. 383-400.

[DH] Diffie, W., and Hellman, M.E., "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[DEK] Dolev, D., Even, S., and Karp, R.M., "On the Security of Ping-Pong Protocols", *Inform. and Control*, Vol. 55, 1982, pp. 57-68.

[DY] Dolev, D., and Yao, A.C., "On the Security of Public-Key Protocols", *IEEE Trans. on Inform. Theory*, Vol. IT-29, 1983, pp. 198-208.

[E] Even, S., "On the Complexity of Some Word Problems that Arise in Testing the Security of Protocols", presented in *NATO Advanced Research Workshop on Combinatorial Algorithms on Words*, Maratea, Italy, June 1984.

[EG] Even, S., and Goldreich, O., "On the Security of Multi-Party Ping-Pong Protocols", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*, 1983, pp. 34-39.

[EGL] Even, S., Goldreich, O., and Lempel, A., "A Randomized Protocol for Signing Contracts", *Comm. of the ACM*, Vol. 28, No. 6, pp. 637-647, 1985.

[GHY] Galil Z., Haber S., and Yung M., "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*,

[GMR] Goldwasser, S., Micali, S., and Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems", *Proc. of the 17th ACM Symp. on Theory of Computation*, 1985, pp. 291-304.

[LMR] Luby, M., Micali, S., and Rackoff, C., "How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin", *Proc. of the 24th IEEE Symp. on Foundation of Computer Science*, 1983, pp. 11-21.

[NS] Needham, R.M., and Schroeder, M.D., "Using Encryption for Authentication in Large Networks of Computers", *Comm. of the ACM*, Vol. 21, No. 12, 1978, pp. 993-999.

[RSA] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Comm. of the ACM*, Vol. 21, February 1978, pp. 120-126.

[R] Rosen, B.K., "Tree-Manipulation Systems and Church-Rosser Theorems", *Jour. of the ACM*, Vol. 20, No. 1, January 1973, pp. 160-187.