A Framework for the Study of Cryptographic Protocols

Richard Berger (*)

Sampath Kannan (**)

René Peralta (***)

Computer Science Division

University of California

Berkeley, California.

ABSTRACT

We develop a simple model of computation under which to study the meaning of cryptographic protocol and security. We define a protocol as a mathematical object and security as a possible property of this object. Having formalized the concept of a secure protocol we study its general properties. We back up our contention that the model is reasonable by solving some well known cryptography problems within the framework of the model.

1. Introduction.

It can be argued that cryptographers have been able to provide satisfactory solutions to only the simplest among the problems involving transactions between mutually suspicious parties. In this category lie problems like flipping coins [1], exchange of a single bit [2] (or a fraction of a bit [3]), demonstrating the truth of some boolean predicates on the secret keys [4], and the Oblivious Transfer [5] [6]. Harder problems

(*) Research sponsored in part by GTE fellowship. (**) Research sponsored by the Helen and George Pardee Fellowship (***) Research sponsored in part by NSF grant MCS-82-04506 and by Universidad Católica de Chile.

H.C. Williams (Ed.): Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 87-103, 1986.

which in our opinion have not been completely solved include exchange of secret keys [7], certified mail [8], contract signing [9], and mental poker [10] [11]. The published solutions to the latter problems either have not been proven secure or use the cryptographic definition of one-way function. Cryptographers use the term one-way function to mean a function which has whatever it takes to make its use in cryptographic protocols secure. In particular, (cryptographic) one-way functions reveal no partial information about their inverse value. Even though one-way functions are useful theoretical objects, the actual encryption functions available in the literature are not one-way functions in this strong sense. Even the probabilistic encryption methods of Blum, Blum, and Shub [12] [13] have not been shown secure under multiple encryptions of the same message or of functionally related messages. It is not clear that there exists secure solutions for the harder problems mentioned above which assume only the hardness of inverting an encryption function. If these solutions do exist, it is likely that considerably more powerful theoretical tools will have to be developed before they can be found. The development of such tools is the objective of this research.

In this paper, we define a cryptographic protocol as a mathematical object and security as a property of this object. Having formalized the concept of a secure protocol, we study its general properties. One of our main motivations for this work is the problem of combinations of protocols. It has been implicitly assumed in the literature that if two protocols are secure then these protocols can be performed sequentially without loss of security. This assumption turns out to be false more often than not. For example the seemingly harmless act of encrypting the same message using Rabin's encryption function under two different composite numbers is insecure. The message can be retrieved in polynomial time from the two encryptions [14]. The use of RSA with small exponent has similar problems [15]. In our model we are able to show a class of secure protocols which is closed under sequential execution. We call protocols in this class **strongly secure.**

Finally, we provide strongly secure solutions to some of the simpler problems discussed above. Solutions to more complex problems typically use these simpler protocols as subroutines. For example the coin flipping protocol is used in practically all solutions to the mental poker problem. Our results, while leaving open the problem of finding secure solutions to the harder class of transaction problems, increases our confidence in the use of simpler protocols as subroutines to more complex protocols.

2. The Protocol Environment or Model of Computation.

We think of a protocol as occurring between two Probabilistic Turing Machines (PTM's) A and B which operate synchronously. Each PTM has, besides its computation tape, a one-way infinite tape for incoming messages. We call this tape the "mailbox" of the machine. The PTM's communicate by writing into each others mailbox (Fig. 1).

We call such a system a CPTM (for Communicating Probabilistic Turing Machines). The mailbox tape symbols are digits , unary minus , letters, punctuation marks , and an end-of-message marker.

The PTMs have the capacity of reading a symbol from its mailbox at the same time as the symbol is being written. This convention is not essential to the model and any of the common resolutions of concurrent write-read will do.

CPTMs satisfy an "independence condition" which is stated as follows:

Independence Condition for CPTMs.

For all j, the conditional distribution of A's (B's) j^{th} message given the prior messages between A and B is independent of the state of B (A) at the time of the message.

The Independence Condition holds for CPTMs because all communication in a CPTM occurs via mailboxes.

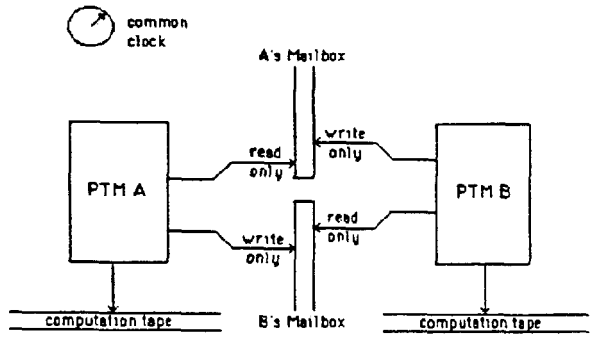

Figure 1. Communicating Probabilistic Turing Machines.

Our model will assume that factoring large integers is hard:

Definition - A **Blum Integer** is a composite $N = P \cdot Q$ where both P and Q are congruent to 3 mod 4 and the length of P is equal to the length of Q.

Assumption - Factoring Blum Integers is hard: For every poly-time probabilistic Turing Machine M, and any polynomial p, the probability that machine M factors a random n-bit Blum Integer is asymptotically less than $\frac{1}{p(n)}$.

3. Protocols Under the Assumption that Factoring Large Integers is Hard.

We now turn to the study of CPTM's whose parts A and B have computed and interchanged keys N_A and N_B with the following properties:

(let N be the key)

i) The Jacobi symbol $(\frac{-1}{N}) = 1$.
ii) N has exactly two distinct prime factors, both odd.
iii) if z is a quadratic residue modulo N , then there exists roots x and y of z with opposite Jacobi symbols.

From now on when we refer to a number being a (public) key, we mean a number having properties i-iii. Under the assumption that factoring Blum integers is hard, A and B can generate (factored) keys whose factorization can not be computed by the opposite party. We will later show that there exist protocols such that the parties to the CPTM can convince each other that N_A and N_B are keys without helping each other factor the key. Note that not all numbers satisfying properties i-iii are Blum integers. We choose this

definition of public key because we know of no secure protocol by which the parties can convince each other that N_A and N_B are Blum integers. On the other hand these properties are enough to solve the problems typically solved by protocols using Blum integers.

4. Initialization of a CPTM.

The initial input to both A and B is a positive integer n, called the "security parameter" of the CPTM. The following steps are then carried on:

Step 1 - A computes a key N_A of length n and writes it on B's mailbox.
Step 2 - B reads N_A from its mailbox.
Step 3 - B computes a key N_B of length n and writes it on A's mailbox.
Step 4 - A reads N_B from its mailbox.

We start counting steps after initialization is completed. i.e. when we say the k-th step we mean the k-th click of the common clock after initialization.

We can not assume that A and B follow the initialization protocol. However, we will assume N_A,N_B are odd, composite, have length n, and satisfy property i) above. We can safely assume these properties because they are verifiable in polynomial time by a PTM without access to the factorization of N_A,N_B. We will later exhibit protocols through which A (B) can prove to B (A) that properties ii) and iii) hold without helping the opponent factor N_A (N_B).

5. The Definition of Cryptographic Protocol.

The concept of cryptographic protocol is used in various ways in the literature. The most common use of the term refers to two or more programs or computers with various communication capabilities. An alternative definition, proposed in [16], [17] , [18] and [19] considers a protocol as a sequence of operators on messages. We will consider only 2-party protocols in which the parties are mutually distrusting. Our definition of cryptographic protocol refers not to the machines executing a communication but to the rules of such interaction. We will also ignore the problems of saboteurs or eavesdroppers on communication lines.

Definition - A protocol Π is a pair [L,t(n)] ($L \in N$, t(n) a polynomially bounded function of n) and two sets of predicates

$P_i^A(m_1^A, \cdots, m_i^A, m_1^B, \cdots, m_{i-1}^B)$
$P_i^B(m_1^A, \cdots, m_i^A, m_1^B, \cdots, m_i^B)$ for $i = 1,\ldots,L$.

We denote the sequences $\{P_i^A\}$, $\{P_i^B\}$ by $\vec{P}^A$,$\vec{P}^B$ respectively. L and t(n) are called the "length" of Π and the "time between messages" of Π respectively . The semantics of this definition is as follows: m_i^A is the state of B's mailbox at time $(2i-1)\cdot t(n)$; m_i^B is the state of A's mailbox at time $2i\cdot t(n)$. It is the responsibility of A to see that P_i^A is true for all i . It is the responsibility of B to see that P_i^B is true for all i .

The sequence $(m_1^A, m_1^B, \cdots, m_L^A, m_L^B)$ is called a **conversation** between A and B. The reason we define m_i^A, m_i^B as states of mailboxes rather than as messages is that the former is always defined whereas the latter may not exist if one of the parties does not follow the protocol. As a consequence of this definition we may define the probability distribution φ_n of $(m_1^A, \cdots, m_L^A, m_1^B, \cdots, m_L^B)$ for fixed PTM's A and B.

We will typically leave t(n) unspecified and argue simply that all computations necessary between messages can be done in probabilistic polynomial time. If A and B are fixed PTM's and Π is a protocol we denote the ordered triple (Π,A,B) by Π(A,B).

Let the symbol Λ stand for logical AND. We define the predicates $P^A(m_1{}^A, \cdots, m_L{}^A, m_1{}^B, \cdots, m_L{}^B)$, $P^B(m_1{}^A, \cdots, m_L{}^A, m_1{}^B, \cdots, m_L{}^B)$ as $\bigwedge_i^L P_i{}^A$ and $\bigwedge_i^L P_i{}^B$ respectively.

Protocol 1: Verifying that N_A satisfies property iii : if z is a quadratic residue modulo N_A , then there exists roots x and y of z with opposite Jacobi symbols.

Blum [1] proposed the following protocol :

For i := 1 to 100

1. A sends to B a random quadratic residue $x_i \ mod \ N_A$.
2. B sends to A $b_i = 1 \ or \ -1$ at random.
3. A sends to B a root of $x_i \ mod \ N_A$ with Jacobi symbol b_i.

If N_A satisfies property iii , then A will always be able to perform step 3. Otherwise , the probability that A can always respond at step 3 is $\leq 2^{-100}$.

In our formalism this protocol is written as follows :

Π_1: Do 100 times the following protocol

$P_1^A = (m_1^A \in Z_{N_A})$

$P_1^B = (m_1^B \in \{-1,1\})$

$P_2^A = (\text{if } P_1^B \ then \ (m_2^A)^2 \equiv m_1^A (\ mod \ N_A) \ \text{ and } \ (\frac{m_2^A}{N_A}) = m_1^B)$

$P_2^B = (m_2^B = \text{"thanks"})$

Notice that this protocol does not tell B how to behave in order to obtain proof that N_A has property iii. (Whereas Blum's version explicitly states how the parties should choose their messages). However, we can show that there exists a poly-time PTM B which follows the protocol such that, after the protocol, either A has been caught cheating , or the probability that N_A satisfies iii is $\geq 1 - 2^{-100}$. *Throughout this paper we take the position that a protocol merely allows the parties to behave so as to achieve the desired goal, it cannot force them to do so.*

6. Security.

We must first develop some notation.

Definition - A **key-generator** with input n generates a random factored n-bit Blum integer.

Definition - A poly-time PTM A is **an honest player** for protocol Π if:

1) its first step is a call to a key generator which returns (P_A, Q_A)

2) it goes through the initialization process with $N_A = P_A \cdot Q_A$

3) for all poly-time PTM B the probability of $P^A(m_1^A, \cdots, m_L^A, m_1^B, \cdots, m_L^B)$ is 1.

Since PTM's have limited computational power it is possible that there exists no honest player for a particular protocol. This motivates the following definition :

Definition - A protocol Π is **A-feasible** if there exists an honest player A for Π. We define **B-feasible** similarly. A protocol is **feasible** if it is A-feasible and B-feasible.

Definition - We say a protocol is **A-secure** if there exists an honest player A for Π such that for all poly-time PTM B the probability that B factors N_A goes to zero with n. The definition of **B-security** is analogous. We say a protocol is **secure** if it is both A-secure and B-secure.

We now define the notion of a **simulatable** player. This notion is essential for proving security of protocols. We want to put in precise terms the intuitive notion that if machine B can simulate the behavior of machine A in the protocol , then it is not possible for A to have released enough information for B to factor A's key.

There are a number of alternative definitions for the notion of simulator. Do we allow a machine which is simulating A to look at B's coin-tosses? Precisely what is to be simulated? In [4] (henceforth called the GMR model), simulators are considered which simply attempt to produce a sequence of messages with the same probability distribution as the actual conversation between A and B. That is, no attempt is made in that model to duplicate the environment in which a conversation between A and B takes place. For example, if B sends a random quadratic residue $x \ mod \ N_A$ to A, and A replies with a random square root of x modulo N_A, then it is easy to produce conversations with the same probability distribution as the actual conversation between A and B. To do this a machine M simply computes a random number x modulo N_A and lets $x^2 \ mod \ N_A$ be the message from B to A and x be the message from A to B. In the GMR model A is said to **release 0 knowledge to B.** On the other hand, we know that A has a chance of at least $\frac{1}{2}$ of releasing the factorization of N_A in this protocol (this is Rabin's Oblivious Transfer Protocol [8]). This awkward problem in the GMR model has no further consequence since, in that model as in this one, we are really interested in machines A which release no information to any machine B. For example, suppose B' is as B above except that it sends A the factorization of N_A if it obtains it. Then it is clear that there is no machine M which simulates A against B' unless factoring is in RP. Hence A does release knowledge to B', even though it does not release knowledge to B. In the GMR model machine A is said to **release 0 knowledge** if for all machines B, it releases 0 knowledge to B.

More serious drawbacks of the definitions in the GMR model are that i) it does not seem to go beyond the obvious statement that A releases no knowledge if and only if no machine B can put knowledge in the communication tape after a conversation (and hence it does not seem to provide a tool for the construction of 0-knowledge protocols) ; and ii) it is not clear whether or not the sequential execution of a polynomial

number of 0-knowledge protocols is still 0-knowledge (concatenation of protocols is a major goal in this model).

We will require that a simulator for machine A against machine B not only produce a possible conversation with the same distribution as conversations between A and B, but that it does so **while duplicating the interaction that B has with A.** It would be too restrictive to require a simulator to do this **all the time** , since it seems that in that case a simulatable machine A could not make use of the factorization of its key. Thus we relax this condition by requiring that a simulator succeeds in simulating A **with a constant probability greater than** 0. In addition to this we must require that a simulator realizes whether or not it succeeded in simulating A, otherwise simulatable protocols turn out to be unconcatenatable. We now formalize these definitions.

Let Π be a protocol and B a player. Let A be an honest player which generates a random n-bit Blum integer N_A for a key. Recall that an honest player A will have put m_i^A in B's mailbox by time $(2i-1)\cdot t(n)$. Let S be a procedure which, when called by B at time $(2i-1)\cdot t(n)$ returns a message m_i^S. Let B[S] be machine B except that at step $(2i-1)\cdot t(n)$ (after initialization) of B, B[S] calls S and sets $m_i^A = m_i^S$. We also give B[S] some extra power as follows: at any time B[S] may return to an earlier configuration an re-start the computation from there. However, we will require that B[S] run in random polynomial time. We also require that S and B satisfy the Independence Condition for CPTMs defined in section 2. Notice that B[S] is a poly-time PTM with input n, N_A. Thus, if N_A is a random n-bit Blum integer, the probability that B[S] can factor N_A is asymptotically 0 by assumption.

We define σ_n to be the probability distribution of $(m_1^S,\cdots,m_L^S,m_1^B,\cdots,m_L^B)$ in B[S] with security parameter n. Recall that φ_n is the distribution of conversations between A and B with security parameter n.

Definition - Let Π be a protocol with A an honest player. We say S is an **A-simulator** for Π if for n sufficiently large, for all players B, and for all pairs (N_A,N_B), machine B[S] satisfies the following conditions:

i) the probability ρ of $P^A(Y)$ given N_A,N_B is
a constant greater than 0 and the event $P^A(Y)$
is independent of B's coin tosses.

ii) S decides $P^A(x)$ with error probability 0 for all x.

iii) $\sigma_n(x \mid P^A(Y),N_A,N_B) = \varphi_n(x \mid N_A,N_B)$ for all x.

where Y is a random variable which assumes values $(m_1^S,\cdots,m_L^S,m_1^B,\cdots,m_L^B)$ in B[S].

Since B[S] can return to earlier configurations we see that the constant ρ can be made exponentially close to 1. For example if $\rho = \frac{1}{2}$ for machine B[S] we can define another machine B'[S] which runs B[S] and if $P^A(Y)$ is not true, runs B[S] again. The probability that $P^A(Y)$ is true for B'[S] is now $\frac{3}{4}$. In general, if we allow for k trials of B[S], the probability of $P^A(Y)$ is $1-(\frac{1}{2})^k$.

Definition - Let Π be a protocol and A an honest player for Π. We say A is **simulatable** if there exists an A-simulator for A.

Theorem 1 . Let Π be a protocol and A an honest player for Π. If A is simulatable and S is a simulator for A then for any pair of keys (N_A,N_B), the probability that B factors N_A given (N_A,N_B) is less than a constant times the probability that B[S] factors N_A. The constant is independent of the keys.

Proof: Fix A , B , N_A , N_B. Let S be an A-simulator for $\Pi(A,B)$. Let X , Y be random variables which assume values $(m_1^A, \cdots ,m_L^A,m_1^B, \cdots ,m_L^B)$ in $\Pi(A,B)$, and $(m_1^S, \cdots ,m_L^S,m_1^B, \cdots ,m_L^B)$ in B[S] respectively. Let E be the event that B factors N_A in Π. Let E' be the event that B[S] factors N_A. For the remainder of the proof all probabilities are conditional on the values of N_A,N_B.

Let Ω be the message space and $x = (m_1^A, \cdots ,m_L^A,m_1^B, \cdots ,m_L^B) \in \Omega^{2L}$. Recall $\varphi_n(x) = \sigma_n(x|P^A(Y))$ and note that the independence condition on S implies that for all x, $Prob.(E|X = x) = Prob.(E \mid Y = x)$.

Thus,

$$
\begin{aligned}
& Prob.(E) \\
&= \int_x \varphi_n(x) \cdot Prob.(E \mid X = x) \\
&= \int_x \sigma_n(x|P^A(Y)) \cdot Prob.(E' \mid Y = x) \\
&= \int_x \sigma_n(x|P^A(Y)) \cdot Prob.(E' \mid (Y = x) \wedge P^A(Y)) \\
&= Prob.(E' \mid P^A(Y)) \\
&\leq \frac{Prob.(E')}{Prob.(P^A(Y))} \\
&\leq \frac{Prob.(E')}{\rho}. \quad \nabla
\end{aligned}
$$

The third equality is justified by the fact that if $P^A(Y)$ is true then $Y = x$ implies $P^A(Y)$.

Corollary 1 . Let Π be a protocol and A an honest player for Π. If A is simulatable then Π is A-secure.

Simulatability is a strong requirement. It is conceivable that a protocol is secure without being simulatable. This motivates the following definition :

Definition - A protocol Π is **strongly secure** if there exist honest simulatable players A and B for Π.

Strongly secure protocols have the desirable property that they release no partial information about the factorization of the player's private keys. This results in strongly secure protocols being "concatenable".That is, a polynomial number of strongly secure protocols can be run under the same keys. We now formalize this idea.

Definition - Let A_1,A_2 be honest players for protocols Π_1,Π_2 respectively. The machine $A_3 = A_1|A_2$ is defined by the following rules :

> A_3 runs as A_1 until A_1 halts.
> Then A_3 runs as A_2 except that, rather
> than obtaining N_A from the key generator, it skips
> the initialization routine using the keys N_A,N_B
> known to A_1 instead.

Definition - Let $\Pi_1 = (L_1,t_1(n),\vec{P}^{A_1},\vec{P}^{B_1})$ and $\Pi_2 = (L_2,t_2(n),\vec{P}^{A_2},\vec{P}^{B_2})$ be two protocols. We define the **concatenation** $\Pi_1 \,\%\, \Pi_2$ of Π_1,Π_2 as follows:

$\Pi_1 \% \Pi_2 = (L_3, t_3(n), \bar{P}^{A_3}, \bar{P}^{B_3})$
where

$L_3 = L_1 + L_2$
$t_3(n) = \max\{t_1(n), t_2(n)\}$

$P_i^{A_3} = P_i^{A_1}$ for $i = 1, \ldots, L_1$
$P_{L_1+i}^{A_3} = P_i^{A_2}$ for $i = 1, \cdots L_2$.
$P_i^{B_3} = P_i^{B_1}$ for $i = 1, \ldots, L_1$

$P_{L_1+i}^{B_3} = P_i^{B_2}$ for $i = 1, \cdots L_2$.

In other words, concatenation of two protocols is simply the concatenation of the two sequences of predicates.

Now we are ready to show that simulatable protocols are concatenable. Even though the statement is intuitively true, the proof is somewhat technical.

Figure 2.a) depicts a CPTM composed of a PTM B and an adversary $A_1|A_2$ where A_1, A_2 are honest simulatable players for two protocols Π_1, Π_2 respectively. A machine R_B (the "restriction of B to Π_1) is defined from machine B in the following way: R_B behaves as B until A_2 starts executing, at which point it halts. Figure 2.b) depicts the same PTM B but with A_1, A_2 replaced by simulators S_1, S_2 respectively. Figure 2.c) depicts PTM B with adversaries S_1 and then A_2. The random variables $X_1, X_2, V_1, V_2, Z_1, Z_2$ represent the messages between $A_1 - B$, $A_2 - B$, $S_1 - B$, $S_2 - B$, $S_1 - B$, $A_2 - B$ respectively in the given configurations.

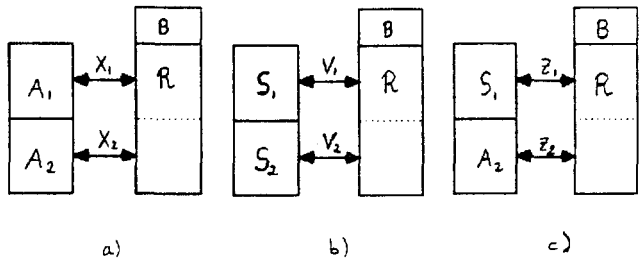

Figure 2.

Theorem 2 . Let Π_1, Π_2 be A-secure protocols with honest simulatable players A_1, A_2. Then $A_3 = A_1|A_2$ is an honest simulatable A-player for $\Pi_3 = \Pi_1 \% \Pi_2$.

Proof: Honesty of $A_1|A_2$ follows immediately from the construction of $A_1|A_2$. Thus we need only show simulatability. We must show that for n large enough, for all PTM B, for all a, b, N_A, N_B,

$$Prob.(X_1 = a, X_2 = b) = Prob.(V_1 = a, V_2 = b \mid P^{A_1}(V_1) \wedge P^{A_2}(V_2)) \quad (*) .$$

Let n, the size of the public keys, be large enough so that the simulatability conditions hold for both A_1 and A_2 with simulators S_1 and S_2 respectively. Fix a, b, N_A, N_B. From now on all probabilities are taken conditioning on the values of N_A, N_B. If $\neg P^{A_1}(a)$ or $\neg P^{A_2}(b)$ then both sides of (*) are zero. Suppose $P^{A_1}(a)$ and $P^{A_2}(b)$. If $P^{A_1}(a)$ and $Prob.(V_1 = a) = 0$ then, by the simulatability conditions, both sides of (*) are 0.

Suppose $Prob.(V_1 = \alpha) > 0$.

We first show that $Prob.(X_1 = \alpha, X_2 = b) = Prob.(Z_1 = \alpha, Z_2 = b \mid P^{A_1}(Z_1))$. Note that this is implied by the 2 equations

(i) $Prob.(X_2 = b \mid X_1 = \alpha) = Prob.(Z_2 = b \mid Z_1 = \alpha)$

and

(ii) $Prob.(X_1 = \alpha) = Prob.(Z_1 = \alpha \mid P^{A_1}(Z_1))$.

The first equation holds by the Independence Condition for CPTMs. Equation (ii) holds by simulatability.

Now we show that

$Prob.(Z_1 = \alpha, Z_2 = b \mid P^{A_1}(Z_1)) = Prob.(V_1 = \alpha, V_2 = b \mid P^{A_1}(V_1) \Lambda P^{A_2}(V_2))$ (**).

This equation is implied by the two equations

(iii) $Prob.(Z_1 = \alpha \mid P^{A_1}(Z_1)) = Prob.(V_1 = \alpha \mid P^{A_1}(V_1) \Lambda P^{A_2}(V_2))$

(iv) $Prob.(Z_2 = b \mid Z_1 = \alpha) = Prob.(V_2 = b \mid P^{A_2}(V_2) \Lambda V_1 = \alpha)$.

Equation (iii) can be shown as follows:

$$Prob.(V_1 = \alpha \mid P^{A_1}(V_1) \Lambda P^{A_2}(V_2)) = \frac{Prob.(V_1 = \alpha \Lambda P^{A_1}(V_1) \mid P^{A_2}(V_2))}{Prob.(P^{A_1}(V_1) \mid P^{A_2}(V_2))}$$

$$= \frac{Prob.(V_1 = \alpha \Lambda P^{A_1}(V_1))}{Prob.(P^{A_1}(V_1))}$$

$$= \frac{Prob.(V_1 = \alpha)}{Prob.(P^{A_1}(V_1))}$$

$$= Prob.(V_1 = \alpha \mid P^{A_1}(V_1)).$$

The second equality holds because the event $P^{A_2}(V_2)$ is independent of the events $(V_1 = \alpha \Lambda P^{A_1}(V_1))$ and $P^{A_1}(V_1)$ by definition of simulatability. The third and fourth equalities hold because we have assumed $P^{A_1}(\alpha)$.

Equation (iv) holds because S_2 is a simulator for A_2 and for all machines B. In particular, S_2 is a simulator for A_2 executing the protocol against a machine M which is $B[S_1]$ with the condition that M chooses it's coin tosses randomly and uniformly only among those which yield $Z_1 = \alpha$. Machine M is depicted in Figures 3.a) and 3.b) playing against A_2 and S_2 respectively. Note that

$Prob.(V_2 = b \mid P^{A_2}(V_2) \Lambda V_1 = \alpha) = Prob.(U_2 = b \mid P^{A_2}(U_2)) = Prob.(T_2 = b) = Prob.(Z_2 = b \mid Z_1 = \alpha)$.

Combining equations (*) and (**) completes the proof of the theorem .Q.E.D.

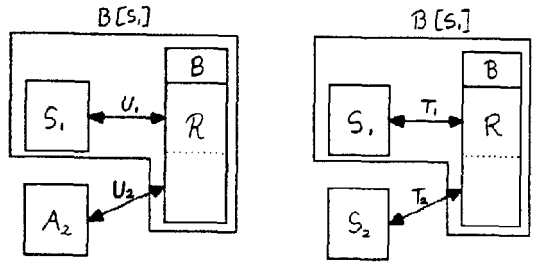

Figure 3.

Theorem 3. Protocol 1 is a strongly secure protocol.

Proof: Protocol 1 is a concatenation of 100 protocols

Π_i: $P_1^A = (m_1^A \in Z_{N_A})$

$P_1^B = (m_1^B \in \{-1,1\})$

$P_2^A = (\text{if } P_1^B \text{ then } (m_2^A)^2 = m_1^A (\text{mod } N_A) \text{ and } (\frac{m_2^A}{N_A}) = m_1^B)$

$P_2^B = (m_2^B = \text{"thanks"})$

Notice that Bob does not use his key, therefore we need only be concerned about the protocol being A-secure. By theorem 2, it is enough to display a simulatable algorithm for A in Π. Let A's algorithm be the following:

message 1:
Send a random quadratic residue $m_1^A \in Z_{N_A}$.

message 2:
If m_1^B is 1 or -1 then send the root of m_1^A with Jacobi symbol m_1^B else send "you are cheating".

Then the following algorithm is an A-simulator:

message 1:
Choose a random number $x \in Z_{N_A}$ and send $m_1^S = x^2 \text{ mod } N_A$.

message 2:
if m_1^B is 1 or -1 and the Jacobi symbol of x modulo N_A is m_1^B then send $m_2^S = x \text{ mod } N_A$. If m_1^B is not 1 or -1 then flip a fair coin. If the outcome is heads send m_2^S = "you are cheating". If the outcome is tails, machine B[S] returns to the initial configuration and the simulation is repeated.

The last instruction of machine B[S] may seem intriguing at first glance. However, it is necessary in order to satisfy properties i) and iii) of a simulator. The problem is the following: if B does not send $m_1^B = \pm 1$ then S has a chance of only $\frac{1}{2}$ of satisfying P^A. Therefore, if S simply responds "you are

cheating" when B sends $m_1^B \neq \pm 1$, the conditional probability of a conversation x given that S satisfies P^A is biased towards those conversations in which B sends $m_1^B \neq \pm 1$.

If we consider B's messages as questions to A (or S) then the problem is that all questions are easily answered by A, whereas the probability that S can answer questions may not be the same for all questions. Machine B[S] must incorporate instructions to homogeneize the hardness of replying to B.

Having said this, verification that B[S] satisfies properties i) to iii) of a simulator is easy and is left to the reader.

7. Strongly Secure Solutions to Some Cryptographic Problems.

In this section we provide strongly secure solutions to some well-known cryptographic problems.

Protocol 2 - Coin Flipping Into a Well.

The purpose of this protocol is for Alice to give Bob a random bit. However, Alice must not know which bit she gave him until Bob displays the bit. Bob, on his part, cannot lie about which bit he got. Since we have shown that Protocol 1 is strongly secure we may assume that N_A satisfies property iii of a public key. The protocol is as follows:

Π_2 : $P_1^A(m_1^A)$ = ("let's flip a coin into your well")
$P_1^B(m_1^B) = (m_1^B \in Z_{N_A})$
$P_2^A(m_2^A) = (m_2^A \in \{1,-1\})$
$P_2^B(m_2^B) = ((m_2^B)^2 = m_1^B \ mod \ N_A)$

Alice's and Bob's algorithms are as follows:

Alice :
message 1:
send m_1^A = "let's flip a coin into your well"

message 2:
send m_2^A = +1 or -1 at random.

Bob :
message 1 :
choose x at random and send $m_1^B = x^2 \ mod \ N_A$.

message 2 :
send $m_2^B = x$.

The value of the coin-flip is $(\frac{x}{N_A})\cdot b$. Bob may display the value of the coin-flip by displaying the root of x^2 that he knows. Until he displays the coin-flip at message 2 (m_2^B), Alice has no idea of what the value of the coin-flip is. If Alice is honest she can be sure that Bob cannot lie about the bit he got because he knows at most one root x of $x^2 \ mod \ N_A$ (and, of course, -x).

This protocol is simulatable because neither party uses the factorization of their keys. Assuming that Bob knows a root x , the probability that the coin-flip is 1 is the same as the probability that the coin-flip is -1. If Bob is honest, he can be sure that the coin-flip is unbiased because, from Alice's point of view, the

probability that x has Jacobi symbol 1 is the same as the probability that x has Jacobi symbol -1.

An important observation, which will be used in protocol Π_4, is that a simulator S can apriori choose the value of the coin-flip provided it chooses it at random. To do this S sends $m_2^S = \pm 1$ at random. If the outcome of the coin-flip is different from the one S wants, machine B[S] can simply restart computation at the beginning of the protocol execution. In random polynomial time, S can obtain the flip it originally chose.

Protocol 3 - Generating a random element in Z_{N_A}.

The coin-flipping Protocol 2 can be used $n = |N_A|$ times to generate a random element in Z_{N_A}.

Protocol 4 - Verifying that N_A has exactly 2 prime factors, both odd.

This problem has been studied extensively by mathematicians. To this date, an efficient algorithm to determine the number of distinct prime factors of a composite number (the index of the number) has not been found. It is possible that no such algorithm exists. It is a remarkable achievement of the research on interactive proof systems that a proof that the index of a composite number can be shown to be 2 by an omniscient party without releasing any additional information about the composite number.

The crucial observation for this solution is due to Adleman [20]. He suggested using the fact that if N_A has more than two prime factors, then at most $\frac{1}{8}$ of the numbers in $Z_{N_A}^*$ are quadratic residues. The protocol uses Protocol 3 to generate M random numbers with Jacobi symbol 1 in Z_{N_A}. Having done this, Alice reveals a square root of each of the numbers which is a quadratic residue. Bob accepts the number N_A if Alice reveals at least αM square roots. The parameters M and α are chosen so as to obtain a negligible probability of error at the minimum possible cost M. This solution has a two-sided error probability. It is possible for Alice to convince Bob that N_A has at most 2 prime factors when it in fact has more than 2, and it is possible for Alice to be unable to complete the proof even though N_A has exactly two prime factors. We now derive an approximation to the optimum value of α.

Let Y be the number of quadratic residues among M random numbers modulo N_A. Let p be the probability that a random number in Z_{N_A} is a quadratic residue. By the Central Limit Theorem (see any probability textbook, for example [21]) the random variable

$$Z = \frac{Y - pM}{\sqrt{Mp(1-p)}}$$

is asymptotically $\Phi(0,1)$ (normal with mean zero and standard deviation 1).

For M in the hundreds and $p \in [\frac{1}{8}, \frac{1}{4}]$, $\Phi(0,1)$ is a good approximation to the distribution of Z. From now on we compute probabilities under the assumption that Z has distribution $\Phi(0,1)$.

Let ϵ_1 be the probability that Bob rejects N_A when N_A has exactly two prime factors. Let ϵ_2 be the probability that Bob accepts N_A when N_A has exactly three prime factors. Then, if N_A has exactly two prime factors we have

$$\varepsilon_1 = P(Y < \alpha M)$$

$$= P(Z < \frac{(\alpha - .25)M}{\sqrt{M(.25)(.75)}})$$

$$= P(Z < \frac{(4\alpha - 1)\sqrt{M}}{\sqrt{3}})$$

$$= \int_{-\infty}^{(4\alpha - 1)\sqrt{M} / \sqrt{3}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}\, dt \qquad \text{(I)}$$

Similarly, if N_A has exactly three prime factors we have

$$\varepsilon_2 = 1 - P(Z < \frac{(8\alpha - 1)\sqrt{M}}{\sqrt{7}})$$

$$= \int_{-\infty}^{-(8\alpha - 1)\sqrt{M} / \sqrt{7}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}\, dt \qquad (II)$$

Lemma 1.

If **x** is negative then $\int_{-\infty}^{x} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}\, dt < \frac{-2e^{\frac{-x^2}{2}}}{x\sqrt{2\pi}}$.

Proof: $x < 0$ and $t \leq x$ implies $-t^2 \leq -xt$.

Thus $\int_{-\infty}^{x} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}\, dt < \int_{-\infty}^{x} \frac{e^{\frac{-xt}{2}}}{\sqrt{2\pi}}\, dt = \frac{-2e^{\frac{-x^2}{2}}}{x\sqrt{2\pi}}$. ▽

Theorem 4. For all values of α, $e^{\frac{-M}{74}} < Max\{\varepsilon_1,\varepsilon_2\} < e^{\frac{-M}{75}}$ asymptotically. We can achieve an error probability in this range if we let $\alpha = \frac{\sqrt{21} - 1}{20}$.

Proof: Since we seek to minimize $Max\{\varepsilon_1,\varepsilon_2\}$, it is clear that the optimal value of α is somewhere between $\frac{1}{8}$ and $\frac{1}{4}$. Thus $(4\alpha - 1) < 0$. Using the lemma , and substituting in $(4\alpha - 1)\sqrt{M}/\sqrt{3}$ and $-(8\alpha - 1)\sqrt{M}/\sqrt{7}$ for **x** in (I) and (II) respectively, we get

$$\varepsilon_1 < \frac{-2\sqrt{3}\, e^{\frac{-(4\alpha - 1)^2 M}{6}}}{(4\alpha - 1)\sqrt{2\pi M}}$$

and

$$\varepsilon_2 < \frac{2\sqrt{7}\, e^{\frac{-(8\alpha - 1)^2 M}{14}}}{(8\alpha - 1)\sqrt{2\pi M}} .$$

Thus both ε_1 and ε_2 are bounded above by functions of the form $a\frac{e^{-bM}}{\sqrt{M}}$. Since the parameter b dominates the expression for the bound , we would like b to be the same for ε_1 and ε_2 , i.e. we want $\frac{(4\alpha - 1)^2}{6} = \frac{(8\alpha - 1)^2}{14}$. Solving for α we get $\alpha = \frac{\sqrt{21} - 1}{20}$, which makes $b \approx .01339 > \frac{1}{75}$. Thus $Max\{\varepsilon_1,\varepsilon_2\} < e^{\frac{-M}{75}}$.

A similar argument , involving the (asymptotic) inequality $e^{-\frac{t^2}{2}} > -t\, e^{-\frac{t^2}{(2-\beta)}}$ for $2 > \beta > 0$ and $t < 0$ shows that MAX$\{\varepsilon_1,\varepsilon_2\}$ is asymptotically greater than $e^{\frac{-M}{74}}$ *Q.E.D.*

We have shown that, even though, this protocol achieves exponentially small probability of error, we must use M in the thousands in order to achieve truly negligible probability of error.

This protocol requires the communication of a very large number of bits. It is expensive in communication and computation. This is also the fastest known protocol for this problem. Goldwasser, Micali, and Rackoff [4] have an elegant but expensive 0-knowledge interactive proof by which Bob can prove to Alice that he knows a root of a quadratic residue modulo her key. Using this technique a 0-knowledge protocol for this problem can be constructed which is essentially a hundred times as expensive as our protocol. Protocols for harder problems, e.g. Blum's certified mail protocol, may require the execution of this protocol hundreds or thousands of times for different keys. This illustrates the practical need for protocols which use a single key.

It would be straight-forward but cumbersome to write this protocol in our formalism. Instead, we write it out in a hybrid notation and argue informally that it is simulatable.

Π_4: Do 3000 times

i) Execute Protocol Π_3 to generate a random number x in Z_{N_A}.

ii) If the Jacobi symbol of $x \ mod \ N_A$ is 1 then Alice sends the message "non-residue" or a square root of $x \ mod \ N_A$.

Theorem 5 . Π_4 is strongly secure.

Proof: The reason that this needs to be proven is that it does not follow immediately from Theorem 2. This is because Π_4 **is not a concatenation of strongly secure protocols.** However, if Alice follows the algorithm given for Π_2 and honestly executes instruction ii) of Π_4 then we can argue that Alice is simulatable.

We argue informally as follows: A simulator for Π_3, the protocol which generates a random element in Z_{N_A}, can choose apriori what number is to be generated (see the note on this matter in the description of Π_2) provided it chooses it at random. Thus S can simulate A as follows: i) S flips a fair coin to decide whether the number generated in the simulation of Π_3 will have Jacobi symbol 1 or -1. If the number is to have Jacobi symbol -1 then S simply generates a random element with Jacobi symbol -1. If the number is to have Jacobi symbol 1 then S flips a fair coin to decide whether it will choose a quadratic residue or a quadratic non-residue. Then S generates a random element in $r \in Z_{N_A}$. If x in step i) of Π_4 is to be a non-residue then S sets $x = -r^2 \ mod \ N_A$. If x is to be a residue then S sets $x = r^2 \ mod \ N_A$. The reader can verify that x, chosen in this way, is indeed a random element in Z_{N_A}. If x is a quadratic residue then S knows a square root of x and thus can execute step ii) of Π_4. *Q.E.D.*

Note that the properties of public key N_A are crucial in this proof. This is because if N_A is a public key then -1 modulo N_A is a non-residue with Jacobi symbol 1. If N_A was an arbitrary composite then this protocol would not be simulatable since there is no known effective algorithm to compute a non-residue with

Jacobi symbol 1 modulo an arbitrary composite. This completes the proof that Alice and Bob can convince each other that $N_A N_B$ are valid public keys without helping the opponent factor the key.

Protocol 5 - The Oblivious Transfer.

A strongly secure variant of Rabin's Oblivious Transfer, called "The Probabilistic Channel", has been implemented in [22] based on an earlier work on the Oblivious Transfer [6].

Acknowledgements.

The importance of studying cryptographic protocols in a rigorous way was made clear to us by Manuel Blum. He also guided us throughout this research with insight and valuable references. He, of course, bears no responsibility for the possible weaknesses and shortcomings of this model. Other good friends who worked with us on this problem include Tom Tedrick and Umesh Vazirani.

References.

1. M. Blum, Coin Flipping by Telephone, *Proc. IEEE COMPCON*, 1982, 133-137.
2. M. Luby, S. Micali and C. Rackoff, How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin., *24th. IEEE Annual Symp. on Foundations of Computer Science*, 1983, 11.
3. T. Tedrick, How to Exchange Half a Bit, *Proceedings of Crypto 83*, N.Y., 1984, 147.
4. S. Goldwasser, S. Micali and C. Rackoff, The Knowledge Complexity of Interactive Proof Systems, *17th. Annual ACM Symp. on Theory of Computing*, 1985.
5. M. Fischer, S. Micali and C. Rackoff, A Secure Protocol for the Oblivious Transfer, *Proceedings of Eurocrypt 84.*, 1984.
6. R. Berger, R. Peralta and T. Tedrick, *A Provably Secure Oblivious Transfer*, Dept. EECS, Univ. of California, Berkeley , Calif. , 1983.
7. M. Blum, How to Exchange Secret Keys, *ACM Transactions on Computer Systems 1*, 2 (May 1983), 175-193.
8. M. Blum, *Three Applications of the Oblivious Transfer : 1. Coin Flipping by Telephone, 2. How to Exchange Secrets , 3. How to Send Certified Electronic Mail*, Dept. EECS, Univ. of California, Berkeley , Calif. , 1981.
9. S. Even, O. Goldreich and A. Lempel, A Randomized Protocol for Signing Contracts, *Technical Report #233*, February 1982..
10. S. Fortune and M. Merritt, Poker Protocols, *Crypto 84*, 1984.
11. M. Yung, Cryptoprotocols : Subscription to a Public Key, the Secret Blocking and the Multi-Player Mental Poker Game., *Crypto 84*, 1984.
12. L. Blum, M. Blum and M. Shub, A Simple Secure Pseudo-Random Number Generator, *CRYPTO 82*, 1982.
13. S. Goldwasser and M. Blum, An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information., *Crypto 84*, 1984.

14. M. Blum, *A Potential Danger with Low-Exponent Modular Encryption Schemes: Avoid Encrypting Exactly the Same Message to Several People.*, U.C. Berkeley Computer Science Department, 1984.

15. J. Hastad, *On Using RSA with Low Exponent in a Public Key Network.*, MIT Computer Science Department, 1984.

16. D. Dolev, S. Even and R. Karp, On The Security Of Ping-Pong Protocols, *Proceedings of Crypto 82*, 1982.

17. D. Dolev and A. Yao, On The Security Of Public Key Protocols, *IEEE Transactions on Information Theory. IT-30* (March 1983), 198.

18. M. Merritt, *Cryptographic Protocols* , Ph.D Thesis. Georgia Institute of Technology, GIT-ICS-83/06. 1983.

19. M. Merritt and P. Wolper, States of Knowledge in Cryptographic Protocols., *Unpublished Manuscript.*, .

20. L. Adleman, Private Communication through M. Blum., 1983.

21. K. Chung, *A Course in Probability Theory*, Academic Press, London, 1974.

22. R. Peralta and T. Tedrick, The Probabilistic Channel, *In preparation*, 1985.