Why and How to establish a Private Code On a Public Network

(Extended Abstract)

Shafi Goldwasser* Silvio Micali** and Po Tong†
University of California, Berkeley

Abstract

The Diffie and Hellman model of a Public Key Cryptosystem has received much attention as a way to provide secure network communication.

In this paper, we show that the original Diffie and Hellman model does not guarantee security against other users in the system. It is shown how **users**, which are more powerful adversarys than the traditionally considered passive eavesdroppers, can decrypt other users messages, in implementations of Public Key Cryptosystem using the RSA function, the Rabin function and the Goldwasser&Micali scheme.

This weakness depends on the bit security of the encryption function. For the RSA(Rabin) function we show that computing, from the cyphertext, specific bits of the cleartext, is polynomially equivalent to inverting the function(factoring). As for many message spaces, this bit can be easily found out by communicating, the system is insecure.

We present a modification of the Diffie and Hellman model of a Public-Key Cryptosystem, and one concrete implementation of the modified model. For this implementation, the difficulty of extracting partial information about clear text messages from their encoding, by eavesdroppers, users or by Chosen Cyphertext Attacks is **proved equivalent** to the computational difficulty of factoring. Such equivalence proof holds in a very strong **probabilistic** sense and for any message space.

No additional assumptions, such as the **existence** of a perfect signature scheme, or a trusted authentication center, are made.

1. Introduction

Diffie and Hellman introduced in [5] the model of a Public Key Cryptosystem for solving the following problem in a communication network:

> "Establishing A private conversation, between two individuals regardless of whether they have ever communicated before."[5]

Briefly, we describe their model,

1.1 The Diffie and Hellman model

Let M be a finite message space, and A, B.. users in the network. Every user, C, publicizes an encryption function $E_C:M \rightarrow M$ in a public file and keeps secret a decryption function D_C such that $D_C(E_C(m))=M$ for $m \in M$. E_C is a trapdoor function; easy to evaluate, but hard to invert unless some secret is known. The secret is C's private information. When user B wants to send a message $m \in M$ to C, he sends $E_C(m)$. All users in the system send messages to C using E_C.

This model is by definition secure against a passive eavesdropper; an adversary who knows the encryption algorithm, can read and store the cyphertext by tapping the lines, and whose objective is to get the cleartext. However, their model is not as secure against more sophisticated eavesdroppers such as

1) **Users** adversarys who can read, store, and send encrypted messages. They have publicized a legal encryption algorithm in the public file, and thus we must assume that they are always capable of sending and receiving messages from all other users in the system. They may also try to impersonate other users.

2) **Chosen Cyphertext Attacks** adversarys who can read and store messages, and have use of the decoding equipment. Thus, they can produce polynomial number of chosen cyphertext pairs of (cyphertext, cleartext).

Yao and Dolev in [15], show that PKC protocols which are secure against a passive eavesdropper, are insecure even in presence of saboteurs; users whose behavior is restricted to applying certain well defined operators to the binary strings they exchange.

We discuss two weaknesses of the PKC model in presence of arbitrary behavior on the part of users and cyphertext attacks.

1.2 The Insecurity of the PKC Model

1) Users may compromise the security of encrypted messages they receive, by replying to them. For many message spaces, from the response to a given message m, it is possible to extract information about m itself. This can be exploited by eavesdroppers who are also *users*, to extract infor-

* This work was supported in part by NSF grant MCS 82-04506 and NSF grant ECS 8110684
** This work was supported in part by NSF grant MCS 82-04506
† This work was supported in part by NSF grant MCS-78-20054

0272-5428/82/0000/0134$00.75 © 1982 IEEE

mation about encrypted messages exchanged in the past between a different pair of users. When this partial information is such that iterative extraction of it, results in a polynomial time procedure to invert the cyphertext or break the system, then legal users in the system may "illegally" decrypt other users messages, and at some cases discover the decryption algorithm of other users.

Examples of the first weakness will be given for different implementations of the Diffie and Hellman model using the RSA function, the Rabin's function and the Probabilistic Encryption scheme of Goldwasser and Micali. In order to do this, we show that for the RSA and the Rabin function, the computational difficulty of computing some fixed bits of x from $f(x)$ is polynomially equivalent to computing all of x from $f(x)$.

2) Users security may be totally compromised if an adversary can momentarily use their decoding equipment The encryption algorithm of user A, E_A, could be such that an adversary who uses A's decoding equipment to compute $D_A(x)$ for a few x's of his choice, may thereafter be able to compute $D_A(x)$ for all x.

The chosen cyphertext attack is a well known threat to cryptographic security. While Rabin's function is known to be vulnerable to it, no one knows how to apply this attack to the RSA or Goldwasser&Micali scheme. However, no encryption scheme has been **proved** secure against such an attack.

1.3 The Modified Model And An Implementation Of It

We propose to modify the Diffie and Hellman model in the following way: the public information is used only to establish a **private code** in between every pair of users wishing to communicate. The private code between user A and B must be established in such a way, so that both users A and B are convinced(receive proof) that they have established the code with each other. To achieve this, A must be able to authenticate a message m to B; that is prove to B that he sent m.

The notion of identity is defined through the public file. More specifically, we propose that every user A puts in the public file a composite integer N_A, whose factorization he keeps private. To be identified as user A when establishing the private code PC, means to prove that the same user who agreed to adopting PC as a code, also knows the factorization of N_A.

We prove that succeeding in being identified as another user in the system, even for a small percentage of the time, is as hard as factoring. We show that the computational difficulty of factoring N_A remains unchanged after A has established a polynomial (in $|N_A|$) number of Private Codes.

Establishing a private code, and subsequently exchanging **authenticated** messages using the private code, makes it impossible for a *user*, unlike in the Diffie and Hellman model, to decrypt messages intended for other users.

The next question considered is what kind of private code should be used? In [8] it is shown that encryption using trapdoor function can not guarantee security of partial information. A probabilistic encryption scheme is proposed in [8], for which, based on the computational difficulty of quadratic residuosity, all partial information about the cleartext is hidden by the cyphertext. In this scheme, however, the bit expansion is quadratic.

Due to recent results by Blum and Micali [3], and Yao[16] on Pseudo Random Bit generation, a new private encoding scheme is possible for which the ratio between *bits sent* and *bits of information* is 1. This scheme maintains the Probabilistic nature of Probabilistic Encryption, and it can be proved that no Partial Information about cleartext messages can be computed from their encoding. We propose an implementation of a pseudo random bit generator, based on factoring. Using this generator we can append authentication tags to encrypted messages, which prevent schosen cyphertext attacks.

The assumption made in this paper is.

Assumption: Let $0<\varepsilon<1$. For each positive integer k, let $C_{k,\varepsilon}$ be the minimum size of circuits C that factor n for a fraction ε of the k bit integers n, which are products of two primes, each $\approx\sqrt{n}$. Then, for every $0<\varepsilon<1$ and every polynomial Q, there exists $\delta_{\varepsilon,Q}$ such that $k>\delta_{\varepsilon,Q}$ implies $C_{\varepsilon,k} > Q(k)$

Related Work

Establishing a private code over the public network has been considered previously ([10], [12]) as usually private codes are more efficient. However, in this paper, we show that they **must** be introduced to guarantee security. In [10], Merkle describes a method to exchange secret keys over insecure channels. His solution is based on the assumption that a "strong" encryption function exists. Needham and Schroeder[11] assume the existence of a central key distribution center.

Table Of Contents

All the results in this paper are based on the assumption that factoring is hard. Section 2 contains background material. Section 3 shows that the computational difficulty of extracting certain cleartext bits, from RSA cyphertext, is equivalent to inverting the RSA function. Similarly, the security of certain boolean predicates over the quadratic residues is shown equivalent to factoring. Section 4 shows how users can decrypt other users messages, for some message spaces and given encryption functions. Examples for encryption schemes suggested in [8], [13], and [14]. Section 5 proposes a protocol for exchanging an authenticated message over the network, which uses a provably secure coin flip protocol. Section 6 uses the protocol developed in section 5, to establish a private code as secure as factoring. An implementation of a Psuedo Random Bit Generator which is as secure as factoring is given in section 6. Section 7 lists open problems.

2. The RSA scheme, the Rabin scheme and Probabilistic Encryption

In order to give concrete examples for problems in PKC model in section 4, let us review the number theoretic implementations of a PKC.

The symbol (x,N) will denote the greatest common divisor of x and N. The symbol (x/N) will denote the Jacobi symbol of x mod N. We let $Z_N^* = \{ x \mid 1 \le x \le N-1$ and $(x,N)=1\}$, and $A_N^* = \{ x \mid 1 \le x \le N-1$ and $(x,N)=1\}$. Given $q \in A_N^*$, deciding whether q is a quadratic residue, is called the quadratic residuosity problem.

2.1 The RSA scheme

In the RSA scheme [14], user A selects N the product of two large primes p_1 and p_2 and a number s such that $(s, \varphi(N)) = 1$, where φ is Euler's totient function. A puts N and s in a public file and keeps the factorization of N private. For message $m \in Z_N^*$, $E_A(m) = m^s$ mod N. No efficient way is known to take sth roots mod N when the factorization of N in unknown.

2.2 The Rabin scheme [13]

The Rabin scheme is a modification of the RSA scheme. In it $s=2$, and thus for all users A, $E_A(x)=x^2$ mod N. Notice that E_A is a 4-1 function as our N is the product of two primes. In fact every quadratic residue mod N, i.e. every q such that $q=x^2$ mod N for some $x \in Z_N^*$ has four square roots mod N: ±x mod N and ±y mod N. As A knows the factorization of N, upon receiving the encrypted message m^2 mod N, he easily computes its four square roots and gets the message m.

2.3 The Probabilistic Encryption Scheme suggested in [8]

In the Goldwasser&Micali scheme, each user in the system publicizes N - the product of two large primes, and y - a quadratic non-residue mod N with Jacobi symbol equal to 1. To send A, a binary message $m=(m_{1,\ldots,}m_k)$, user B randomly picks $x_i \in Z_N^*$ and sends $(e_1, \ldots, e_k)$ to A, where $e_i=x_i^2$ mod N if $m_i = 0$ and $e_i=yx_i^2$ mod N otherwise. Thus every 0 in the message will be represented by a quadratic residue and every 1 by a quadratic non residue with Jacobi symbol 1. A, who knows the factorization of N can easily distinguish residues from non residues mod N and decode the message.

3. The Security of specific bits for the RSA encryption function, and for the Rabin encryption function

Let $f: X \to Y$ be a trapdoor function, $n = |y|$, $y \in Y$. Then, for some $1 \le i \le n$, it is computationally infeasible, from $f(x)$, to compute the ith bit of x for all $x \in X$. The above argument is non-constructive. For the RSA function and the Rabin function, we are able to show something stronger: we identify several such bits, and prove that they are as hard to compute as inverting the function. We exhibit boolean predicates B, $B: Z_N^* \to \{0,1\}$ such that the ability to compute $B(x)$, from $f(x)$, for each $x \in Z_N^*$ is polynomially equivalent to inverting f, i.e there exist a polynomial time algorithm with oracle B, that computes x from $f(x)$, for all $x \in Z_N^*$.

In particular, inverting the RSA is equivalent to: Given x^s mod N

1. Computing the parity of x.
2. Computing whether $x \le \frac{N}{2}$.
3. When $N=b_n \cdots b_l \cdots b_1$, where $b_l=\ldots=b_1=1$ and $b_{l+1}=0$, computing any of the l least significant bits.

Similarly, We show that the difficulty of factoring is equivalent to: Given x^2 mod N,

1. Computing whether the parity of y = parity of x where $x,y \le \frac{N}{2}$, and $x^2=y^2$ mod N.
2. Computing the parity of y such that $y \le \frac{N}{2}$, $(y/N)=1$ and $x^2=y^2$ mod N.

3.1 The security of specific bits for the RSA encryption function

Let N and s be public keys in the RSA scheme. Let $|N| = n$. For any integer m, let $B(m)$ be the binary representation of m, i.e. $B(m) = m_{n-1}m_{n-2} \cdots m_0$ where $m_i = 0$ or 1 and $m = \sum_{i=0}^{n-1} m_i 2^i$. For any binary sequence $t = t_{k-1}t_{k-2} \cdots t_0$, let $M(t)$ be the integer $\sum_{i=0}^{k-1} t_i 2^i$. Let t and u be two binary sequences such that $|u| \le |t|$. Define $t\ (-)\ u$ to be the binary sequence v of length $|u|$ such that $M(t) - M(u)$ has binary representation $w*v$ for some w (we use * for concatenation of binary sequences), i.e. v consists of the last $|u|$ bits of the binary representation of $M(t) - M(u)$.

Example 1 Let the oracle O_1 be defined by

$$O_1(m^s \mathit{mod}\ N) = \begin{cases} 0 & \text{if } m \text{ is even} \\ 1 & \text{if } m \text{ is odd} \end{cases}$$

for m in Z_N^*. Then m can be retrieved from its encryption $m^s \mathit{mod}\ N$ by means of n calls to the oracle O_1. This is achieved by the following algorithm.

Algorithm 1

```
Input: $m^s mod\ N$ in $Z_N^*$
Output: $m$
(1)    Compute $I$ in $Z_N^*$ such that $2^s I\ mod\ N = 1$;
(2)    $r[1] := m^s mod\ N$;
       $ANS[1] := O_1(r[1])$;
       for $i = 2$ to $n$ do
           if $ANS[i-1] = 0$ then $r[i] := r[i-1]I\ mod\ N$
           else $r[i] := (N-r[i-1])I\ mod\ N$;
           $ANS[i] := O_1(r[i])$;
(3)    $t^{(n)} := ANS[n]$;
       for $i = n$ downto $2$ do
           if $ANS[i-1] = 0$ then $t^{(i-1)} := t^{(i)} * 0$
           else $t^{(i-1)} := B(N)\ (-)\ (t^{(i)} * 0)$;
(4)    $m := M(t^{(1)})$;
       Output $m$.
```

Proof that Algorithm 1 works

Let $u[i]$ in Z_N^* be such that $u[i]^s \mathit{mod}\ N = r[i]$. Let $v^{(i)} = B(u[i])$. We shall prove the claim:

$v^{(i)} = w^{(i)} * t^{(i)}$ for *some* $w^{(i)}$

In particular $v^{(1)} = w^{(1)} * t^{(1)}$ for some $w^{(1)}$. But $|v^{(1)}| = |t^{(1)}| = n$, hence $v^{(1)} = t^{(1)}$ and $m = u[1] = M(t^{(1)})$. To prove the claim we use induction. Clearly the claim is true when $i = n$. Now suppose $v^{(i)} = w^{(i)} * t^{(i)}$. For the case $ANS[i-1] = 0$, since $r[i] = r[i-1]I \ mod \ N$, so

$$r[i-1] = r[i]2^s \ mod \ N = (2u[i])^s mod \ N$$

and hence $u[i-1] = 2u[i]$. It follows that

$$v^{(i-1)} = v^{(i)}*0 = w^{(i)} * t^{(i)} * 0 = w^{(i-1)} * t^{(i-1)}$$

For the cases $ANS[i-1] = 1$, since s is odd, so

$$r[i-1] = (N-r[i]2^s) \ mod \ N = (N-(2u[i])^s) \ mod \ N$$

$$= (N-2u[i])^s \ mod \ N$$

and hence $u[i-1] = N-2u[i]$. It follows that

$$v^{(i-1)} = w^{(i-1)} * (B(N) \ (-) \ (t^{(i)} * 0)) = w^{(i-1)} * t^{(i-1)}$$

An example of how Algorithm 1 works follows. Set $N = 37\times31 = 1127$, $s = 7$. In this case $I = 690$. The input to the algorithm is $m^s mod \ N = 81$. $B(N) = 10001111011$ and $n = 11$. The following table describes what happens in the algorithm.

i	$r[i]$	$ANS[i]$	$t^{(i)}$
1	81	1	00001101011
2	313	0	1000001000
3	334	0	100000100
4	1060	0	10000010
5	761	1	1000001
6	236	1	011101
7	34	1	01111
8	627	0	0110
9	211	1	011
10	79	0	00
11	601	0	0

$m = M(t^{(1)}) = 107.$

Example 2 Let the oracle O_2 be defined by

$$O_2(m^s mod \ N) = \begin{cases} 0 \text{ if } m < N/2 \\ 1 \text{ if } m > N/2 \end{cases}$$

for $m^s mod \ N$ in Z_N^*. Then m can be retrieved from its encryption $m^s mod \ N$ by means of n calls to the oracle O_2.

Proof The two oracles O_1 and O_2 are in fact equivalent in the sense that one can be used to simulate the other. For any z in Z_N^*, $2z \ mod \ N$ is even iff $z < \frac{N}{2}$. Hence for any x in Z_N^*, $O_1(2^s x \ mod \ N) = 0$ iff $O_2(x) = 0$. This shows O_1 can be used to simulate O_2. Recall $I = 2^{-s} mod \ N$. For any y in Z_N^*, $O_1(y) = 0$ iff $O_2(Iy) = 0$. This shows O_2 can be conversely used to simulate O_1.

Is every bit of x as hard to compute as inverting $f(x)$? We can show that if N ends in a sequence of k 1's, then each one of the k least significant bits of x is as hard to compute, from $f(x)$, as inverting $f(x)$.

Example 3 Let k be a fixed integer such that $0 \le k < n$ and the last $k+1$ bits in $B(N)$ are all 1's. let the oracle O_3 be defined by

$$O_3(m^s mod \ N) = m_k$$

where $m^s mod \ N$ is in Z_N^* and $B(m) = m_{n-1}m_{n-2} \cdots m_k \cdots m_1 \cdots m_0$, i.e.$O_3$ returns the $(k+1)$ st least significant bit of $B(m)$. Then m can be retrieved from its encryption $m^s mod \ N$ by means of n calls to the oracle O_3. This is achieved by using the following algorithm.

Algorithm 3

```
Input: $m^s mod \ N$ in $Z_N^*$
Output: m
(1)   Compute $I$ in $Z_N^*$ such that $2^s I \ mod \ N = 1$;
(2)   $f^{(1)}[0] := 0, f^{(1)}[1] := 0, \ldots, f^{(1)}[k-1] := 0$;
(3)   $r[1] := m^s mod \ N$;
      $ANS[1] := f^{(1)}[0]$;
      $f^{(1)}[k] := O_3(r[1])$;
      for $i = 2$ to $n$ do
        if $ANS[i-1] = 0$
        then $r[i] := r[i-1]I \ mod \ N$;
            $f^{(i)}[k-1]f^{(i)}[k-2] \cdots f^{(i)}[0] :=$
            $f^{(i-1)}[k]f^{(i-1)}[k-1] \cdots f^{(i-1)}[1]$
        else $r[i] := (N-r[i])I \ mod \ N$;
            $f^{(i)}[k-1]f^{(i)}[k-2] \cdots f^{(i)}[0] :=$
              the first $k$ bits in
            $B(N) \ (-) \ f^{(i-1)}[k]f^{(i-1)}[k-1] \cdots f^{(i-1)}[0]$
            {comment: $f^{(i)}[j] :=$ complement of
            $f^{(i-1)}[j+1]$ for every $j = 0,1,\ldots,k-1$}
        endif
        $f^{(i)}[k] := O_3(r[i])$;
        $ANS[i] := f^{(i)}[0]$;
(4)   $t^{(n)} := f^{(n)}[k]f^{(n)}[k-1] \cdots f^{(n)}[0]$;
      for $i = n$ downto $k+2$ do
            if $ANS[i-1] = 0$ then $t^{(i-1)} := t^{(i)} * 0$
            else $t^{(i-1)} := B(N) \ (-) \ (t^{(i)} * 0)$;
(5)   $x := 2^k M(t^{(k+1)}) \ mod \ N$;
      if $x^s mod \ N = m^s mod \ N$ then output $m = x$
      else output $m = N-x$.
```

Proof that Algorithm 3 works

Let $u[i]$ be defined as before. We shall prove the following claim (α):

The last $k+1$ bits of $B(u[i])$ are $f^{(i)}[k]f^{(i)}[k-1] \cdots f^{(i)}[0]$ for any $i = k+1, k+2, \ldots, n$.

For the time being suppose that claim (α) is true, then $u[k+1] = M(t^{(k+1)})$ (this follows from the way that the $t^{(i)}$'s are constructed). It follows from definition that $u[k+1] = \pm 2^{-k} m \ mod \ N$, so $m = \pm 2^k M(t^{(k+1)}) \ mod \ N$. To prove claim (α), we show that it is true for the case $i = k+1$. The general claim then follows from induction as in the proof of validity of algorithm 1. The special case $i = k+1$ of claim (α) follows from claim (β):

$f^{(i)}[k]f^{(i)}[k-1] \cdots f^{(i)}[0]$ and the last $k+1$ bits of $B(u[i])$ coincides in the first i bits for any $i = 1,2,\ldots,k+1$.

We now prove claim (β) by induction. It is true for $i = 1$ by the definition of $f^{(1)}[k]$. Now suppose the claim is true for $i-1$. We separate the arguments into cases.

case 1 $f^{(i-1)}$ = the least significant bit of $B(u[i-1])$: Clearly the claim is true for i.

case 2 $f^{(i-1)} = 0$, but the least significant bit of $B(u[i-1]) = 1$:

In this case, $u[i] = \frac{N+u[i-1]}{2}$. Since the last $k+1$ bits in $B(N)$ are all 1's, so the j th least significant bit of $B(u[i])$ coincides with the $(j+1)$ st least significant bit of $B(u[i-1])$ for $j = 1,2,...,k$ and hence the claim is true for i.

case 3 $f^{(i-1)} = 1$, but the least significant bit of $B(u[i-1]) = 0$:

In this case, $u[i] = N - \frac{u[i-1]}{2}$. Again since the last k+1 bits in $B(N)$ are all 1's, so the j-th least significant bit of $B(u[i])$ is different from the $(j+1)$ st least significant bit of $B(u[i-1])$ for $j = 1,2,...,k$ and hence the claim is true for i.

The following example illustrates how Algorithm 3 works. Set $N = 37\times67 = 2479, s = 7$, $k = 3$. In this case $I = 949$. The input to the algorithm is $m^s mod\ N = 328$. $B(N) = 100110101111, n = 12$. The following table describes what happens in the algorithm.

i	$r[i]$	$f^{(i)}[3]$	$f^{(i)}[2]$	$f^{(i)}[1]$	$f^{(i)}[0]$	$t^{(i)}$
1	328	1	0	0	0	...
2	1397	1	1	0	0	...
3	1967	1	1	1	0	...
4	2475	1	1	1	1	000101101111
5	1317	0	0	0	0	10000100000
6	417	0	0	0	0	1000010000
7	1572	1	0	0	0	100001000
8	1949	0	1	0	0	10000100
9	267	0	0	1	0	1000010
10	525	0	0	0	1	100001
11	54	0	1	1	1	00111
12	813	0	1	0	0	0100

$M(t^{(4)}) = 367$, $\pm2^3\times367\ mod\ 2479 = 457\ or\ 2022$. Output $m = 457$.

3.2 The security of specific bits for the Rabin encryption function

We give results similar to those in section 2.1. We follow the same notation as in that section. For convenience, we define

$$parity(x) = \begin{cases} 0 & \text{if } x \text{ is even} \\ 1 & \text{if } x \text{ is odd} \end{cases}$$

for any x in Z_N^*.

Example 4 Let the oracle O_4 be defined by

$$O_4(z) = \begin{cases} 0 & \text{if } parity\ (x) = parity\ (y) \\ 1 & otherwise \end{cases}$$

where z is a quadratic residue in Z_N^* and x,y are the two smaller square roots of z in Z_N^*. Then N can be factored by means of n calls to the oracle O_4. This is achieved by using a similar algorithm to the one in Example 5.

Example 5 Let N be a product of 2 primes p and q both congruent to 3 mod 4. Let the oracle O_5 be defined by

$O_5(z) = parity\ (y)$

where y is the square root of z in Z_N^* such that $y \le \frac{N}{2}$ and y has Jacobi symbol 1. (Due to the special condition p and q, the 2 smaller square roots of z always has different Jacobi symbols.) then N can be factored by means of n calls to the oracle O_5. This is achieved by Algorithm 5.

Algorithm 5

```
(1) Compute $I$ in $Z_N^*$ such that $4I\ mod\ N = 1$;
(2) Pick an integer $x < \frac{N}{2}$ in $Z_N^*$ with Jacobi symbol -1
    at random;
(3) $r[1] := x^2 mod\ N$;
    $ANS[1] := O_5(r[1])$;
    for $i = 2$ to $n$ do
            $r[i] := r[i-1]I\ mod\ N$
            $ANS[i] := O_5(r[i])$;
(4) $t^{(n)} := ANS[n]$;
    for $i = n$ downto 2 do
            if $ANS[i-1] = 0$ then $t^{(i-1)} := t^{(i)} * 0$
            else $t^{(i-1)} := B(N)\ (-)\ (t^{(i)} * 0)$;
(5) $y := M(t^{(1)})$;
    {comment:$y$ is the other smaller square root of $z$
    in $Z_N^*$}
```

Proof that Algorithm 5 works

Let $u[i]$ in Z_N^* be such that $u[i] < \frac{N}{2}$ and has Jacobi symbol 1. Since both p and q are congruent to 3 mod 4, so both 2 and -1 has Jacobi symbol 1 in Z_N^* and hence we have

$u[i-1] = 2u[i]$ if $ANS[i-1] = 0$

and

$u[i-1] = N - 2u[i]$ if $ANS[i-1] = 1$

The other part of the proof is similar to that in algorithm 1.

Remark The above example 5 implies that, there is no polynomial procedure that computes $O_5(x)$ **for all** the $x \in Q$. The natural question is for what fraction of $x \in Q$ is it possible to compute $O_5(x)$ before factoring becomes easy.

We are able to prove

Theorem 1 Let f be a function from the set Q of quadratic residues in Z_N^* to $\{0,1\}$, where N is a product of 2 primes both congruent to 3 mod 4. Let $|N| = n$. Let

$GOOD = \{z \in Q \mid f(z) = O_5(z)\}$

If $\frac{|GOOD|}{|Q|} \ge \frac{1}{n}$, then for any $\delta > 0$, there is an algorithm to factor N with probability of success $\ge 1-\delta$ and running time polynomial in n, $\frac{1}{\delta}$ and the time to compute f.

Proof Let

$$GOOD\ CHAIN = \{\ z \in Q \mid z, \frac{z}{4}, \frac{z}{4^2}, \ldots, \frac{z}{4^{n-1}} \in GOOD\ \}$$

Define $BAD = Q - GOOD$ and $BAD\ CHAIN = Q - GOOD\ CHAIN$. Since each element in BAD can give rise to at most n elements in $BAD\ CHAIN$, so $|BAD\ CHAIN| \le n\ |BAD|$. If

$\frac{|GOOD|}{|Q|} \geq \frac{1}{n}$, then $\frac{|BAD|}{|Q|} < \frac{1}{n}$. This implies $\frac{|BAD|}{|Q|} < \frac{1}{n}(1 - \frac{1}{n^t})$ for some fixed t. Hence $\frac{|BAD\ CHAIN|}{|Q|} < 1 - \frac{1}{n^t}$ which gives $\frac{|GOOD\ CHAIN|}{|Q|} \geq \frac{1}{n^t}$. For any given $\delta > 0$, the following procedure factors N with probability of success $\geq 1-\delta$ using $\frac{n^{2t+1}}{\delta}$ calls to the function f. Simply carry out algorithm 5 with O_5 replaced by f. Since we pick x at random in step 2, the probability that $z = x^2 mod\ N$ lies in *GOOD CHAIN* is $\geq \frac{1}{n^t}$. If this does happen, then algorithm 5 successfully factors N. Otherwise just repeat with another choice of x in step 2. By the weak law of large numbers, if we execute the algorithm $\frac{n^{2t}}{\delta}$ times, the probability that we successfully factor N is $\geq 1-\delta$. QED

4. Public Key Cryptosystems introduce oracles

We are now ready to see, how the results of the previous section endanger the security of the Diffie and Hellman model.

A **user** knows other users public encryption algorithms, can tap the lines, and send or receive messages to and from all other users in the system.

For many message spaces, the answer to a message may reveal information about the message itself. Let us proceed more formally.

Let M= Message space, $E:M \to M$ be the public encryption function. For each $m \in M$, let $\alpha_{A,B}(m) \subset M$ denote the set of possible answers that A gives to B upon receiving message m. So, upon receiving $E_A(m)$ from B, A picks m' in $\alpha_{A,B}(m)$ and sends B, $E_B(m')$. We say that,

Definition 1: M is **one bit revealing** if it can be partitioned

$$M = M_1 \cup M_2,\ M_1 \cap M_2 = \phi$$

and, (4.1)

$$\bigcup_{m \in M_1} \alpha_{A,B}(m) \quad \cap \quad \bigcup_{m \in M_2} \alpha_{A,B}(m) = \phi$$

And it is "easy to test" membership in $\bigcup_{m \in M_1} \alpha_{A,B}(m)$.

The "bit" that $\alpha_{A,B}(m)$ reveals about m is the boolean predicate $B:M \to \{0,1\}$ where $B(m)=1$ if and only if $m \in M_1$

Let $E:M \to M$ be an encryption algorithm in the Diffie and Hellman model, M a one-bit revealing message space, and let the bit B that m reveals be such that there exists a polynomial time algorithm with oracle B, that inverts $E(x)$ for all $x \in Z_N^*$. Then, independently of the difficulty of inverting E, the system (M,E) is insecure against users.

Do such insecure systems exist? Let us see some natural ways to partition the message space, which are fatal to the security of concrete implementations of public key cryptosystems.

4.1 Examples

Let user A in the PKC publicize a trapdoor function $E_A:X \to Y$. Assume that while user B is sending messages to user A using E_A, an error occurs in the transmission. Such an error will result in A receiving a non meaningful string of bits instead of a message. If users in a Public Key Cryptosystem are not passive recipients of messages (a rather unattractive model !), but they are interested in understanding the communication, A should notify B that an error occurred. Thus notice that any user in the system **may check** whether a particular $y \in Y$ is the image under E_A of a legal message or the image of some "garbage". He does this by simply sending y to A and waiting for A's response. Thus, A acts as an oracle that distinguishes encrypted English from encrypted non-English.

If we want Public Key Cryptosystems that are provably secure, the above oracle like anything else that a user cannot obtain by himself (i.e. without the help of the PKC mechanism), should be suspected of being helpful in inverting $E_A(y)$.

Indeed, this oracle affects the Probabilistic Encryption scheme by Goldwasser and Micali.

Example 6: The encrypted English / encrypted garbage oracle is dangerous for Probabilistic Encryption.

Suppose A publicizes (N,y) where y is a quadratic non-residue modulo N whose Jacobi symbol is 1. B has sent A $E_A(m)$ where m is an English message such that $B(m) = m_1, \ldots, m_k$. $E_A(m) = (e_1, \ldots, e_k)$ where e_i is a quadratic non residue picked at random from A_N^* if $m_i = 0$ and e_i is a quadratic residue picked at random from A_N^* otherwise. User C has tapped the line and got hold of $(e_1, \ldots, e_k)$ and now wants to find out what m_s was ($s \in [1,k]$). C forms an English question α such that $B(\alpha) = \alpha_1, \ldots, \alpha_j$. Let T be a subset of the indices i such that $\alpha_i = 0$. T is such that by replacing α_i by 1 for all $i \in T$, the question alpha gets transformed into a meaningless string of zeros and ones. Then

```
    For all $i \in T$
    C picks an element $x \in Z_N^*$ at random
    sets $b_i = x^2 \bmod N$
    end.
And,
    For all $i \notin T$
    C picks an element $x \in Z_N^*$ at random
    if $\alpha_i = 0$
      then $b_i = x^2 e_s \bmod N$
      else $b_i = x^2 \bmod N$
    end.
```

Lastly, C sends to A, Q = $(b_1, \ldots, b_k)$. Clearly, if e_s was a residue mod N then for all $i \in T$, b_i will be a residue mod N, and thus Q is a legal encryption of question M by the Goldwasser&Micali scheme. Otherwise, if e_s was a non residue mod N, then Q is an encryption of a meaningless string of bits. Upon receiving Q, A will either answer C's question or, in case Q became meaningless, ignore it or ask for a new transmission. In the first case B concludes that m_s was a 0, and in the later cases that m_s was a 1.

It is also reasonable to assume that a response to a "yes" is different from a response to a "no". This suggests the next example.

Example 7(RSA): Let E_A be A's RSA encryption fuction. Suppose, to encrypt a "yes" and a "no" one encrypts an even number and an odd number, respectively. (Alternatively, a number less than $N/2$ will denote a "yes", and a number greater than $N/2$ will denote a "no"). It is straight forward to see that O_1 and O_2 (of section 3) can be implemented by means of the PKC mechanism: If B is a user commounicating with A, who wants to decrypt messages previously received by A, or to simulate a cyphertext attack on E_A, he just waits for the first question that A asks him, to find out the parity(or msb) of x for a chosen $y=E_A(x)$.

Another possible set up is the following. Let $M=Z_N^*$. Let users exchange encrypted packets of the form $m \cdot b$ (where $\cdot$ stands for concatenation, $m \in M$) such that $b=0$ if an acknowledgement of receiving the packet is requested and 1 otherwise. To find out what $O_1(m)$ is, B sets $packet=m$, and sends A, $E_A(m)$. If he receives an acknowledgement, m is even.

The problem of a one-bit revealing message space, effects not only the security of Public Key Cryptosystems, but also the security of protocols. The x^2 mod N function is used often in protocol design, for coin flipping, secret exchange protocols etc. We present a protocol for coin flipping which , based on results presented thus far, if played more than once, is insecure. Can you see why?

Example 8 A and B wants to generate a random binary sequence using a coin flipping protocol based on the Rabin encryption function.

Procedure

Step 0 A gives B an integer, product of two large primes, but keeps secret its prime factorization.

Step 1 B picks x in Z_N^* at random and sends $z = x^2 mod\ N$ to A. B sets $x^* = \min(x, N-x)$.

Step 2 A computes $\pm x$ and $\pm y$, the square roots of z. She, A, computes $x^* = \min(x,N-x)$ and $y^* = \min(y,N-y)$ and bets randomly on whether B possesses x^* or y^*. She finds the first integer i such that the i th least significant bit in the binary representation of x^* is different from that of y^* and sends B her bet:

"the i th least significant bit in the binary representation of the minimum of the 2 square roots that you know is 1 (is 0)"

Step 3 B sends x^* to A.

This procedure determines one bit in the random sequence. The bit, say, is 0 iff B wins. When this is played once, B learns only what the ith least significant bit of y is, which can not enable him to factor N (see section 5.2). The procedure is repeated to generate a sequence of desired length. Finally A releases the factorization of N to prove to B that she did not cheat.

In this protocol B can easily implement the oracle O_4 given in section 2.2. Let z be any quadratic residue in Z_N^* , to find $O_4(z)$, B simply sends z to A in step 1. Then $O_4(z) = 0$ iff the integer i declared by A in step 2 is > 1. It follows from example 4 that if the desired sequence has length longer than $n = |N|$, then B can factor N using n executions of the procedure and control the outcome of the remainder of the sequence.

Thus, the protocol designer must be aware, that if protocols are repeated a polynomial number of times, enough partial information may be leaked to endanger the entire system.

4.2 Encryption Functions Which Are Insecure In Presence Of Users

The above examples suggest the following definition. Let $M=\{0,1\}^n$, $E=$ encryption algorithm used.

Definition 2: Let $E:M \to Z$ be a public encryption function. We say that the pair (E,M) introduces a **dangerous oracle** in the presence of users if :

1. There exists a non-constant, boolean predicate $B:M \to \{0,1\}$ such that

2. There exist polynomial time procedures $T:Z \to Z$, $L:M \to M$ such that, for all $x \in M$, there exist $y \in M$, $T(E(|x|))=E(y)$ and $L(x)=y$

3. There exists polynomial time procedure P such that for some polynomial $Q(\,)$, given $B(x)$, $B(L(x))$, ..., $B(L^{Q(|x|)}(x))$, P computes x.

4. There exists M_1,M_2 that partition M as in (4.1) and $x \in M_1$ if and only if $B(x)=1$.

5. How to Modify the PKC Model.

How can the oracles presented in the last section be prevented? One might suggest to

> "Use trapdoor functions f such that for any message space M, (f,M) has no dangerous oracle".

But this is an extraordinary property for f, as M is arbitrarily large, and there are $2^{|M|}$ partitions of M. In example 6 this would amount to not being able to ask for another copy of a message, when the transmission is erroneous.

However, note that the danger presented in section 4 arises because all users in the network send messages to user A using the same public code. Thus one may, by communicating with A , extract information about the code and understand messages other users send A. We propose a different solution. Let every pair of users A and B use a private code, such that to send m to A, B sends m encrypted in the private code plus a proof that m was sent by him.

However, the nicest feature of PKC is that users can communicate without having ever met before. In order to achieve this, based on the assumption that factoring large composite numbers is hard, we show how A and B can exchange keys(S_{AB} and S_{BA}) across the public network so that B proves to A that he sent S_{AB}, and A proves to B that he sent S_{BA}.

5.1 The Scheme

Every user, A, in the system stores his Primary Key in the public file: an integer N_A product of two large primes. When two users A and B wish to communicate, they contact each other in clear text and make use of their primary keys N_A and N_B, to authenticate to each other their secondary keys, respectively, S_{AB} and S_{BA}. The secondary keys are themselves products of two large primes and can be used by A and B to communicate using any of the above mentioned schemes. In case of the Goldwasser&Micali scheme also a quadratic non residue mod N_A and a quadratic non residue mod N_B should be authenticated.

We now describe an outline of a protocol for the authentication of S_{AB} , that has some promising ideas, but unfortunately is incorrect.

AUTH(A , B , S_A)

Step0: A sends S_A to B.
repeat
  Step1: B sends A random number $r \in Z^*_{N_A}$
until $u = S_{AB} r$ is a quadratic residue mod N_A
Step2: A sends B the $\sqrt{u}$ mod N_A.
Step3: B checks that u^2 mod N_A is of the form $S_A r_i$ mod N_A

The above protocol tries to achieve the following goal: if an adversary C is impersonating A, i.e. if C contacted B on the network to establish a private code pretending to be user A, C will not be able to perform Step 2. By lemma 3, the ability of extracting square roots, even for 1% of the quadratic residues mod N_A, is equivalent to factoring N_A.

However, we should, immediately suspect the above protocol of being incorrect. A is publicizing the result of a computation, $\sqrt{u}$ mod N_A, that involves knowledge of his secret, the factorization of N_A. Any computation performed by A, that B can not do himself, should be regarded as helpful to B for discovering A's secret. In fact, it is easy to see that if at step1 B picks $b \in Z^*_{N_A}$ at random and sets $r_i = b^2 S_A^{-1}$ mod N_A, at the end of the protocol he has 50% chances to factor N_A.

A way is needed for A to be sure that the r_i's have been generated at random.

5.2 A Provably Good Coin Flipping Protocol

A main avenue to avoid making stronger assumptions than the intractability of factoring, index finding, deciding quadratic residuosity modulo composite moduli etc. is to introduce a suitable set of atomic protocols simple enough to be fully understood and proved. The atomic protocol needed for this paper is:

Coin Flipping: Users A and B want to produce a number r by exchanging messages over the network so that, if at least one of A and B want it to, every bit of r is equally likely to be 0 or 1.

Coin Flipping Protocol

Step 0: A gives B an integer N, product of two large primes, but keeps secret its prime factorization.

Step 1: B picks $x \in Z^*_N$ at random and sends A, q = x^2 mod N. B sets x^*= min (x, -x mod N).

Step 2: A computes ±x and ±y, the square roots of q. She, A, computes x^* = min (x, -x mod N) and y^* = min (y, -y mod N), and bets randomly on whether B possesses x^* or y^*. She finds the first position $i \in [1, |N|]$ such that the ith bit of x^* is different from the ith bit of y^* and sends B her bet : "the ith bit of the minimum of the 2 square roots that you know is 1 (is 0)".

Step 3: B sends x^* to A.

Step 4: A releases the factorization of N.

Let

$$I \leftarrow \begin{cases} 1 \text{ if B wins} \\ 0 \text{ if B loses} \end{cases}$$

Theorem 2: In the above Protocol, if A selects her bet at random or if B selects $x \in Z^*_N$ at random, then $Prob(\,I=1\,)=\frac{1}{2}$.

Proof: It is clear that, if A selects her bet at random or if B selects $x \in Z^*_N$ at random, A has $\frac{1}{2}$ probability of guessing correctly the ith bit of x^*. Thus for every instance of the Protocol for which B cannot cheat, the event "B wins" is a Bernoulli variable with probability $\frac{1}{2}$.

B can cheat only if, in Step 3, he is able to compute the other square root of q. Thus, by Rabin theorem, he can cheat only if he can factor N. The only help that the protocol gives B for factoring N is telling him that i is the first positive integer such that the ith bit of x^* is different from the ith bit of y^*. But such additional information cannot effectively help in factoring: B can guess i in all the possible |N| ways ! □ (Note that by releasing the factors of N at stage 4, we avoid the problem described in section 4)

Remark: Notice that in the above coin flipping protocol, A and B cannot prove to a third party who won. Moreover either one of them may interrupt the protocol at any moment. However such extra features are not needed, and will not be used in this paper !

5.3 A protocol for authentication of secondary keys

The protocol is essentially the one outlined in section 5.1, except that the r's of step1 are generated by means of the coin flipping protocol in the last section. The proof, now not difficult, of its correctness in a system with only users and passive eavesdroppers will appear in the full paper.

Note that this protocol is not provably secure against a **meddler** who is the most powerful adversary in a network. A **meddler** can alter, delete, delay, and resend the cyphertext crossing the network, in addition to being a user. In order to avoid simulation of a meddler by a user, we assume that messages are received by users in the same order they were sent.

6. Establishing A Private Code

We have seen how, for any message space M, user A *can send an* authenticated message m to user B, where $m \in M$. Using this authentication facility we now show how and what type of private code should be established between A and B.

We would like to establish a code for which: for any message space, no partial information about the cleartext, can be computed from the cyphertext for even ε of the time(as defined in [8]), AND is economical on the number of bits exchanged over the network. Based on new results in [3] and [16]. it is possible to quickly generate a sequence of bits, whose length is polynomial in the length of the seed, in which given the first k bits, it is computationally infeasible to predict the $k+1$st bit in the sequence with any advantage.

We suggest a private code, which posses the desired properties. which is based on an implementation of a psuedo random bit generator (PSRG), whose cryptographical security is equivalent to factoring. Using this private code, the ratio between bits exchanged and bits of information =1.

Let AUTH(A , B , m) stand for the protocol provided in the previous section, for sending m authenticated from A to B.

We now describe the protocol that A and B should follow in order to establish the private code. Let S_{AB} and S_{BA} each be a composite number product of two primes, both congruent to 3 mod 4.

Establishing the code

step 1: AUTH (A , B , S_{AB})
step 2: AUTH (B , A , S_{BA})
step 3: B picks $y \in Z^{*}_{S_{AB}}$ where (y/N) =1 at random.
step 4: AUTH (B, A, y^2 mod S_{AB})
step 5: AUTH (A , B, z mod S_{AB}) where $z^2=y^2$ mod S_{AB} and $(z/N)=-1$
step 6: A sets $seed:=\{x_1,x_2, \ldots, x_n\}$, $E_{seed}:=\{x_1^{2^r}, x_2^{2^r}, \ldots, x_n^{2^r}\}$, where the $\{x_i\}$'s are residues in $Z^{*}_{S_{AB}}$ picked at random, $r=poly(|n|)$.
step 7: AUTH (A, B, E_{seed})
end

Upon termination of step 3 through step 5, A and B both know the factors of S_{AB}. No other user does. They will use S_{AB}, and *seed* from step 6, as input to the the PSRG described below. Note: B can compute the *seed*, since he can factor.

6.1 A Psuedo Random Bit Generator As Cryptographically Secure As Factoring

Let N be a product of 2 primes both congruent to 3 mod 4, $|N| = n$,Q be the set of quadratic residues in Z_N^{*}. Recall that in theorem 1 (section 3.2) we proved that, if factoring is hard, then the fraction of quadratic residues mod N for which it is easy to compute $O_5(x)$ is less than $\frac{1}{n}$. This result combined with ideas of Yao in [16] about using the "xor" function , suggest the following function g.

Let C_m be the collection of all families of m elements in Q. Define the function g from C_m to $\{0,1\}$ by

$$g(\{z_1,z_2, \ldots, z_m\}) = O_5(z_1) \oplus O_5(z_2) \oplus \cdots \oplus O_5(z_m)$$

where $\oplus$ stands for exclusive or.

Theorem 3: For any polynomially computable, function $f: C_m \rightarrow \{0,1\}$ and any $\varepsilon > \frac{1}{n^t}$, for fixed $t>0$, let

$$GOOD = \{\, z \in Q \mid Pr[g(A)=f(A) \mid z \in A] \geq \frac{1}{2} + \varepsilon \,\}$$

If $\frac{|GOOD|}{|Q|} \geq \frac{1}{n}$, then for any $\delta > 0$, there exists a polynomial, in n, ε^{-1}, δ^{-1}, time algorithm which factors N with probability $\geq 1-\delta$.

Proof First notice that if $z \in GOOD$, then by the weak law of large numbers, we can compute $O_5(z)$ by means of $\frac{n}{\varepsilon^2\delta}$ calls to the function f with probability of success $1 - \frac{\delta}{2n}$. This is achieved by the following procedure P.

Procedure P

(1) Compute
$f(\{z_1,z_2, \ldots, z_m\}) \oplus O_5(z_2) \oplus O_5(z_3) \oplus \cdots \oplus O_5(z_m)$
for $\frac{n}{\varepsilon^2\delta}$ different random choices of $y_2,y_3, \ldots, y_m$ such that $y_i^2 \, mod \, N = z_i$ and y_i has Jacobi symbol 1 for $i = 2,3,\ldots,m$.

(2) The value of O_5 is determined by the majority of the outcomes of the computations in (1).

As before let

$$GOOD\ CHAIN = \{\, z \in Q \mid z, \frac{z}{4}, \frac{z}{4^2}, \ldots, \frac{z}{4^{n-1}} \in GOOD \,\}$$

$BAD = Q - GOOD$ and
$BAD\ CHAIN = Q - GOOD\ CHAIN$. Again

$\frac{|GOOD\ CHAIN|}{|Q|} \geq \frac{1}{n^t}$ for some fixed t. The following strategy factors N with probability of success $\geq 1 - \delta$ using $\frac{n^{2t+2}}{\varepsilon^2\delta^2}$ calls to the function f. Carry out algorithm 5 with O_5 replaced by procedure P. If $z = x^2 mod\ N$ picked at step 2 lies in *GOOD CHAIN*, then algorithm 5 successfully factors N with probability $\geq 1 - \frac{\delta}{2}$, using $\frac{n^2}{\varepsilon^2\delta}$ calls to the function f. Otherwise just repeat with another choice of x in step 2. By the weak law of large numbers, if we execute the algorithm $\frac{n^{2t}}{\delta}$ times, the probability that $z = x^2 mod\ N$ lies in *GOOD CHAIN* is $\geq 1 - \frac{\delta}{2}$ and hence the probability that we factor N is $\geq (1 - \frac{\delta}{2})^2 \geq 1 - \delta$. QED

Now, when we choose m suitably, we get that the value of g can not be predicted with the slightest advantage unless the factorization of N is known.

Corollary 4:
Assume that factoring N is hard. Then for any $\varepsilon > \frac{1}{n^t}$, for fixed $t>0$, **any** $m>t$, any polynomial, in n, time computable function $f: C_m \to \{0,1\}$, and any $A \in C_m$ picked at random $Pr[f(A)=g(A)] < \frac{1}{2} + 2\varepsilon$.

Proof Pick $z_1, z_2, \ldots, z_m$ at random. Let $A = \{z_1, z_2, \ldots, z_m\}$. There are two cases:

Case 1$A \cap BAD$ is empty.

By theorem 2, this happens with probability $< (\frac{1}{n})^m$

Case 2$A \cap BAD$ is non-empty.

In this case, there exists $z \in A \cap BAD$. It follows from the definition of BAD that $Pr[f(A)=g(A)] < \frac{1}{2} + \varepsilon$.

Hence $Pr[f(A)=g(A)] < \frac{1}{2} + \varepsilon + (\frac{1}{n})^m < \frac{1}{2} + 2\varepsilon$.

QED

We now use g to implement a cryptographically secure pseudo random bit generator based on the assumption that factoring is hard. The squaring operation done in step (1) is similar to generators in [2] and [4], but the bit we extract is different ! Each bit in our sequence can be generated in time $O(n^2)$.

Let $seed := \{x_1, x_2, \ldots, x_n\}$, where $x_i \in Q$ for all $1 \le i \le n$. Let b be a binary array, $r = poly(n)$

Algorithm 6.1
Input: *seed*, N, factorization of N
Output: a sequence of r bits

```
For $i=0$ to $r$ do
    (1) $b[r-i] := g(x_1^{2^i}, x_2^{2^i}, \ldots, x_m^{2^i})$
(2) output the $r$ bits $b[0], \ldots, b[r]$
end.
```

We now prove that if it is possible to infer the $k+1$st bit of the sequence, after seeing k bits of it, then we have a fast algorithm to factor N.

Claim 5: Let P() a polynomially bounded function. Let $\varepsilon > \frac{1}{n^t}$ for some fixed $t>0$, $m>t$. For all $k \le P(n)$, and for any polynomial computable function $f: \{0,1\}^k \times C_m \to \{0,1\}$, given $b[0], \ldots, b[k]$ generated by algorithm 6.1 and $\{x_1^{2^r}, \ldots, x_m^{2^r}\}$, $Prob(f(b[0],..,b[k], x_1^{2^r},..., x_m^{2^r}) = b[k+1]) < \frac{1}{2} + 2\varepsilon$

Proof: Lets assume, for contradiction, that $Prob(f(b[0],...,b[k], x_1^{2^r},..., x_m^{2^r}) = b[k+1]) \ge \frac{1}{2} + 2\varepsilon$ for some $\varepsilon > \frac{1}{n^t}$, $t>0$, $m>t$, $k \le P(n)$. Pick $y_1, \ldots, y_m$ from C_m at random. Then for $i=1$ to k let $b[k-i+1] = g(y_1^{2^i},..., y_m^{2^i})$. Now note, that $b[k+1] = g(y_1, \ldots, y_m)$. And $Prob(f(b[0],...,b[k], y_1^{2^r},..., y_m^{2^r}) = g(y_1,..., y_m)) > \frac{1}{2} + 2\varepsilon$ which contradicts corollary 4. QED

6.2 How To Encrypt Using the Private Code

Let $seed = \{x_1, x_2, \ldots, x_n\}$. Let $msg\#$ be a counter for the number of messages A has sent B. Let $P(seed) = p_1, p_2, \ldots, p_k$ stand for the polynomial number of bits generated by the PSRG suggested in 6.1 with inputs S_{AB} and *seed*. Then, to send a message m, $B(m) = m_1 m_2 \cdots m_t$ to B, A sends $(msg\#,\ e_1, e_2, \ldots, e_t, p_{msg\#+t}, \ldots, p_{msg\#+n})$ where $e_i = m_i + p_{i+msg\#}$ mod 2. Essentially, A uses the PSRG in order to generate a one time pad. Also, instead of using AUTH to authenticate every message, A just appends to every encrypted message, a subsequence of the pad, whose length is the length of the seed. Similar authentication tags have been suggested in [].

Let $PR_n(C)$ be the probability that user C ≠ A , B successfully understands a single bit of an encrypted message sent from A to B, or forges a single message. Then,
claim 6: Assume factoring is hard. For the protocol described above, given polynomial number of encrypted messages, for any user C ≠A or B, $PR_n(C) \to 0$ as $n \to \infty$

7. Open problems

Notice that the protocol in section 5.3 is insecure if there is an adversary C who is able to alter and remove messages exchanged in the Network. In fact, C can do the following:

Let h_N be an easy to evaluate function known to C. In step0, C captures S_A and removes it from the network. C computes $C_A = h_N(S_A)$ and sends it to B instead of S_A. h_N is such that the factorization of C_A is known to C.

In step1 C lets A and B generate the randome r_i's, without interfering.

In step2, C captures $\sqrt{v}$ mod N_A, he squares it mod N_A to get v. Next, C computes $vS_A^{-1} = r_i$. The C_A sent instead of S_A is such that for any r_i, the knowledege of $\sqrt{S_A r_i}$, allows C to compute $\sqrt{C_A r_i}$ mod N_A. Thus, in step2, C sends $\sqrt{C_A r_i}$, and has authenticated his key.

It may seem that the possibility of computing such a C_A in Step 0 is a bit unlikely. The following example shows how complex the issue of security is in a Public Key Cryptosystem with meddlers.

Example 9: In step0, C intercepts S_A and sends to B, $C_A = S_A^3$ mod N_A.
In step1, C lets the r_i's reach A
In step2, C intercepts $\sqrt{v} = \sqrt{S_A r_i}$ and sends instead $S_A \sqrt{S_A r_i} = \sqrt{S_A^3 r_i}$ mod N_A.

Thus, he has validated S_A^3 mod N_A. Since N_A and S_A are totally independent, S_A^3 mod N_A is basically a random number, and thus (unlike S_A) probably much easier to factor as it will have many small prime factors !

The essence of example 9 is that a meddler, given x and a one-way function f, may quickly compute y_x so that, given $f^{-1}(x)$, it will be easy for him to compute $f^{-1}(y_x)$.

A possible way out is requiring the function f to be a one way function that "randomly" maps X into X. Do such functions exist? We just saw that $f(x)=x^2$ mod N does not have this extra feature.

Is it possible to build a Public Key Cryptosystem provably secure against an adversary which may be an eavesdropper, user or meddler ?

Acknowledgements

We are grateful to Manuel Blum for having taught us how to design and how to suspect cryptographical protocols.

We thank Eric' Bach, Richard Karp, David Lichtenstein, Ron Rivest and Andy Yao for their valuable help.

References

[1] Adleman, L., *On Distinguishing Prime Numbers from Composite Numbers*, Proceedings of the 21st IEEE Symposium on the Foundations of Computer Science (FOCS), Syracuse, N.Y., 1980, 387-408.

[2] Blum, L., Blum, M., Snub, M.,*A Simple Secure Pseudo-Random Number Generator*, CRYPTO 1982.

[3] Blum, M., Micali, S., *On Generating Cryptographyically Strong Bit Sequences with Arbitrarily Small Bias*, FOCS, 1982.

[4] Brassard, G., *On computationally Secure Authentication Tags Requiring Short Secret Shared Keys*,, CRYPTO 1982. Crypto82.

[5] Diffie, W., and M. E. Hellman, *New Direction in Cryptography*, IEEE Trans. on Inform. Th. IT-22, 6 (1976), 644-654.

[6] Fischer, M., *Private Communication*, through David Lichtenstein.

[7] Goldwasser S., and Micali S., *A Bit by Bit Secure Public Key Cryptosystem*, Memorendum NO. UCB/ERL M81/88, University of California, Berkeley, December 1981.

[8] Goldwasser S., and Micali S., *Probabilistic Encryption & How to Play Mental Poker, Keeping Secret All Partial Information*, Proceeding of 14th STOC Conference, San Francisco, Ca., 1982.

[9] Guy K. R., *How to Factor A Number*, Proceedings of Fifth Manitoba Conference On Numerical Math., 1975, pp. 49-89.

[10] Merkle, R., *Secure Communiction Over Insecure Channnels*, Comm. ACM Vol 21, no. 4, pp. 294-299.

[11] Miller, G., *Riemann's Hypothesis and Tests for Primality*, Ph.D. Thesis, U.C. Berkeley, 1975.

[12] Needham, R. M., Schroeder, M. D., *Using encryption for Authentication In Large Networks of Computers*,Comm. ACM 2 (1978), 993-999.

[13] Rabin, M., *Digitalized Signatures and Public-Key Functions As Intractable As Factorization*, MIT/LCS/TR-212, Technical Memo MIT, 1979.

[14] Rivest, R., Shamir, A., Adleman, L., *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, February 1978.

[15] D.Dolev, A.Yao, *On the Security of Public Key Protocols*, Proceedings of the 22nd IEEE Symposium on the Foundations of Computer Science (FOCS), Nashville, Tennessee, 1981, 350-357.

[16] Yao, A., *A Relation Between Random Polynomial Time and Deterministic Polynomial Time*, FOCS 1982.