Fair Exchange of Secrets

(extended abstract)

Tom Tedrick *

Computer Science Division
573 Evans Hall
University of California
Berkeley, California 94720

Electronic mailing address:
tedrick@berkeley

Key Words and Phrases: Exchange of Secret Keys, Contract Signing, Exchange of Secrets, Fractions of a Bit, Oblivious Transfer, Cryptographic Protocols

Abstract

We consider two problems which arose in the context of "The Exchange of Secret Keys" (see [1]).

(1). In the original protocol, one party may halt the exchange and have a 2 to 1 expected time advantage in computing the other party's secret. To solve this problem, when there is a particular point in the exchange where this time advantage may be critical, we presented at CRYPTO 83 (see [5]), a method for exchanging "fractions" of a single bit.

In this paper we extend the method so as to apply it to all bits to be exchanged, and show how it can be used in a more abstract setting (as in [2]).

(2). We also present a solution to the problem of how to ensure a fair exchange of secrets when one party in the exchange is "risk seeking", while the other is "risk-adverse".

Notation:

We use ^ to signify exponentiation, i.e. 2^k represents 2 raised to the kth power.

Introduction:

The following scenario occurs in both [1] and [2].

There are 2 parties A (Alice) and B (Bob).

Alice holds n pairs of m bit long secrets

$< a(1,1), a(1,2) >, < a(2,1), a(2,2) >, \ldots , < a(n,1), a(n,2) >$

Bob knows exactly one secret from each pair, while Alice does not know which one he knows (this condition can be achieved using an "oblivious transfer" protocol as in [2] and [4]).

Similarly Bob has n pairs of secrets denoted

$< b(1,1), b(1,2) >, \ldots , < b(n,1), b(n,2) >$

Alice knows exactly one secret from each pair, etc.

Each party is eager to know both elements of any pair of his counterpart's secrets. In Blum's "Exchange of Secret Keys" such knowledge allows one to factor (i.e. obtain the secret key of an RSA/Rabin or Goldwasser/Micali public key crypto-system). In the Even/Goldreich/Lempel "Randomized Protocol for Signing Contracts" knowledge of a pair constitutes having a signature to the contract.

We assume that computing a secret can only be done by an exhaustive search of the secret space (the set of m-bit long strings).

*Research sponsored in part by the National Science Foundation Grant MCS 82-04506

G.R. Blakley and D. Chaum (Eds.): Advances in Cryptology - CRYPTO '84, LNCS 196, pp. 434-438, 1985.

Both Blum and Even/Goldreich/Lempel apply the following protocol, in order to reach concurrent knowledge of one pair of secrets:

For k=1 to m do

(1). Alice sends the kth bit of each a(i,j)

(2). Bob sends the kth bit of each b(i,j)

Note that in order to prevent the counterpart from getting any pair of secrets, a dishonest party must send incorrect bits for at least one element in each pair. But the chance of getting away with this is $1/2^n$.

If both parties follow the protocol properly each can compute a complete pair of their counterpart's secrets in about the same amount of time. However some problems arise which we discuss in the following sections.

First Problem:

If Bob halts the protocol after Alice sends the kth bits of her secrets then Bob has a 2 to 1 computational advantage. He needs only search through a subset of $2^{(m-k)}$ possible secrets to compute a pair, while Alice needs search a subset of size $2^{(m+1-k)}$ (twice as large).

In [5] we discussed several methods of exchanging fractions of a bit when there is a key bit that is crucial. Micali/Luby/Rackoff have also written a very nice paper on exchanging a secret bit using a different approach (see[3]).

We here extend one of the methods in [5] so as to carry it out throughout the exchange, keeping Bob's computational advantage at any point below a predetermined amount.

The method (an example)

We first illustrate the method by giving an example.

Bob and Alice agree that the maximum computational advantage will be 5 to 4 (instead of 2 to 1 as in the original protocol).

For each a(i,j), Alice stores the strings

000

001

010 *

011

100

101

110

111

Exactly one of these strings corresponds correctly to the first three bits of a(i,j), say 010 (marked with a * for reference).

Bob acts similarly.

Now a series of exchanges takes place.

For each a(i,j) Alice sends the message:

the next three bits of a(i,j) are not xyz (say not 101 for example).

Bob responds similarly.

Note that after Alice sends her messages Bob has first an 8 to 7, then a 7 to 6, then a 6 to 5, then a 5 to 4 edge (in the ratio between the size of the secret space Alice has to search and the space Bob has to search).

When only half the original strings remain, say for example

001

010 *

110

111

8 new strings (for each a(i,j) etc.) are created by adding 0 or 1 to the old strings. We get:

0010

0011

0100

0101 *

1100

1101

1110

1111

Note again that exactly one of these strings corresponds to the correct first 4 bits of a(i,j).

The exchange then takes place again until 4 strings remain (for each a(i,j)), 8 new strings are created, etc.

Note that the maximum computational advantage for Bob is 5 to 4.

Note that the chance of getting caught cheating by sending incorrect strings is exactly the same as in the original protocol: each time incorrect information is sent the chance of being detected is 50%.

A more formal description of the method:

(1). Decide on an acceptable integer k, where the maximum computational advantage will be (2^k)+ 1 to 2^k.

(2). For each a(i,j) and b(i,j) store the 2^(k+ 1) strings of length k+ 1.

(3). Repeat until done:

For x=1 to 2^k do:

Alice sends a string, for each a(i,j)

Bob sends a string, for each b(i,j)

End {For}

Alice and Bob create 2^(k+ 1) strings, for each a(i,j) and b(i,j), by adding 0 or 1 to the 2^k unsent strings.

End {Repeat}

Time/space complexity of the method:

With k chosen as above, there are o(n*(2^k)) strings of length <= m to be stored. So memory needed is o(n*m*(2^k)).

Time needed is o(n*(m^2)*(2^k)). We present later a slightly more complicated version of the method which may require an additional log(m) factor.

Suggested modification of "The Exchange of Secret Keys"

Shamir/Goldreich have announced a method for breaking the original exchange of secret keys protocol. We suggest that the protocol should be modifed in two ways:

(1). In the original protocol a(i,1) and a(i,2) are distinct square roots of some quadratic residue Xi modulo Alice's public key, where Bob chooses Xi. We suggest that Alice should choose Xi at random and send Bob a root via oblivious transfer (see [4]).

(2). In the original protocol Alice sends the bits of a(i,j) in order, first, second, third ... . Instead the location of the next bit to be sent should be chosen randomly from the unused locations. The method described above for sending less than a bit fits well into this type of scheme and will be described briefly by means of an example.

The method modified to be more random looking:

Suppose we set a 5 to 4 advantage limit.

For each a(i,j) we create 8 strings (say the bit length m=20). First choose a random location, say 5. Create 2 strings

- - - - 0 - - - - - - - - - - - - - -

- - - - 1 - - - - - - - - - - - - - -

Note that exactly one of these strings corresponds correctly to the bits of a(i,j).

For each string create 2 new strings by choosing a random location and filling it with 0 or 1.

- - - - 0 - - - - - - - - - 0 - - - - -

- - - - 0 - - - - - - - - - 1 - - - - -

- - - - 1 - - - - - 0 - - - - - - - - -

- - - - 1 - - - - - 1 - - - - - - - - -

Note again that exactly one string corresponds to the real a(i,j).

Repeat the above process for each of the 4 strings:

- - - - 0 - - 0 - - - - - - 0 - - - - -

- - - - 0 - - 1 - - - - - - 0 - - - - -

- - - - 0 - - - - - - - - - 1 - - - 0 -

- - - - 0 - - - - - - - - - 1 - - - 1 -

- - - - 1 - - - - - 0 - - - - - 0 - - -

- - - - 1 - - - - - 0 - - - - - 1 - - -

- - - - 1 - - - - - 1 - 0 - - - - - - -

- - - - 1 - - - - - 1 - 1 - - - - - - -

Again note that exactly one of these strings corresponds correctly to a(i,j).

As in the first version, half the strings are exchanged. Then 8 new strings are created using the 4 remaining strings. For example if the remaining 4 strings were:

- - - - 0 - - 1 - - - - - - 0 - - - - -

- - - - 0 - - - - - - - - - 1 - - - 1 -

- - - - 1 - - - - - 0 - - - - - 0 - - -

- - - - 1 - - - - - 0 - - - - - 1 - - -

we would create 8 new strings by choosing a random location in each old string and filling it with 0 or 1, getting say:

- - - - 0 - - 1 - - - - - - 0 - - - - 0

- - - - 0 - - 1 - - - - - - 0 - - - - 1

- - - - 0 - - - - 0 - - - - 1 - - - 1 -

- - - - 0 - - - - 1 - - - - 1 - - - 1 -

- - - - 1 - 0 - - - 0 - - - - - 0 - - -

- - - - 1 - 1 - - - 0 - - - - - 0 - - -

- - - - 1 - - - - - 0 0 - - - - 1 - - -

- - - - 1 - - - - - 0 1 - - - - 1 - - -

And so on. At each step exactly one string corresponds correctly to a(i,j).

Risk seeking vs. Risk Adverse

We showed that the expected time for computing a secret can be made reasonably equal for both parties. However this may not be enough to discourage "risk-seeking" parties which may try to exploit the fact that the variance is large.

Let Ta and Tb be random variables representing the time Alice and Bob need to compute each other's secret. We assume Ta and Tb have identical uniform distributions on some interval 1 to K. Neglecting insignificant terms (as we will throughout this analysis) we get E(Ta)= K/2.

Let Y=Ta-Tb. Then E(|Y|)= K/3. So there is a good chance that if Bob halts the protocol at a certain point, he will discover Alice's secret well before she discovers his.

Note E(Y^2)= (K^2)/6, or E(Y^2)= 1.5*(E(Ta))^2

One solution to this problem is to modify the nature of the secret. We take a large number, say X, old secrets. The new secret is defined to be knowledge of all X old secrets (there are interesting crypto-systems based on using a large number of keys, see [6], so this idea is not far fetched).

Note that Ta is now the sum of X uniformly distributed random variables. If Y=Ta-Tb as before, we find that E(Y^2)= (1.5*(E(Ta))^2)/X as opposed to 1.5*(E(Ta))^2 in the previous case. So the squared distance Ta-Tb is reduced by a factor of X. So the expected distance between Ta and Tb can be reduced to any level desired.

This is somewhat analagous to flipping a silver dollar as opposed to flipping 100 pennies ... If Alice gets the heads and Bob gets the tails then in each case they expect to get 50 cents, but in the first case the variance is larger.

Overall costs are multiplied by a factor of X if this method is used.

Acknowledgements:

Oded Goldreich made many helpful suggestions.

References:

[1]. Blum, M. "How to Exchange Secret Keys", ACM Transactions on Computer Systems, 1983.

[2]. Even,Goldreich,Lempel "A Randomized Protocol for Signing Contracts"

[3]. Micali,Rackoff,Luby "The MiRackoLus Exchange of a Secret Bit", 1983 FOCS

[4]. Peralta,Berger,Tedrick "A Provably Secure Oblivious Transfer", Eurocrypt 84

[5]. Tedrick,T. "How to Exchange Half a Bit", CRYPTO 83

[6]. Tedrick,T. "Some Advantages of Using Many Keys in Public Key Encryption Protocols"