# A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party

*Catherine Meadows*
Naval Research Laboratory
Code 7593
Washington, D. C. 20375

*ABSTRACT*

The problem of authentication of mutually suspicious parties is one that is becoming more and more important with the proliferation of distributed systems. In this paper we construct a protocol in which users can verify whether they have matching credentials without revealing their credentials to each other unless there is a match. This protocol requires a trusted third party, but does not require it to be available to the users except when they sign up for the system. Thus it is useful in situations in which a trusted third party exists, but is not available to all users at all times.

## 1. Introduction

The problem of authentication of mutually suspicious parties is one that is becoming more and more important with the proliferation of distributed systems. A user in a distributed system may not only need to verify his identity to the system, but may require that the system, or another user or node in the system, verify itself to him. Moreover, both sides may require some degree of authentication before they release any information about themselves. Consider, for example, the problem of two users who wish to discuss classified information, but each of who must know if the other has the appropriate clearance and need to know before he can reveal his own. Another example of the sort of problem that can arise is the example of a high-level job-referral service described by Baldwin and Gramlich.[1] Since a corporation does not want to announce an opening except to people who would be interested in accepting it, and a person interested in such a job does not want it to be generally known that he is planning on leaving his current place of employment, such a job-referral service would be required to give information only to people whose interests match.

The usual solution to such problems is to consult a trusted third party. However, this may not be practical in a highly distributed situation. For one thing, there may not be a central authority that all parties are willing to trust, as in the case of the job-referral service. Or such a central authority may exist, but may not be available to all users at all times.

One solution to these problem lies in the development of *cryptographic matchmaking protocols,* that is, protocols in which users with various secrets can verify whether or not their secrets agree without revealing the secrets to each other or a third party. In 1985 Baldwin and Gramlich[1] devised such a cryptographic matchmaking protocol. Their protocol involves the use of a third party which keeps a list of encrypted secrets and informs users whenever there is a match. In order to establish whether he has a match with another user, a user must establish communication with the third party and the other user. A user is totally responsible for his secrets, that is, he can lie if he wants to, and the only thing preventing him from lying is the possible embarrassment of being believed to have a secret he does not have by anyone who successfully requests a match with him. Such a protocol is consistent with the requirements of a job-referral service, in which the user has the leisure to consult a centralized matchmaker, and in which there is no central authority to verify that someone is actually seeking the job he says he is seeking.

The protocol we will describe below will satisfy somewhat different requirements. Suppose that one or more organizations wish their representatives to be able to exchange secret credentials without revealing the credentials to each other unless there is a match, but that it is not always possible for the representatives to consult their parent organizations, and that it is not possible to construct a third party that will be available to all representatives at all times.

The organizations are able to place a high degree of trust in each other (or else there would be no point in exchanging credentials). They are also willing to place a certain amount of trust in their representatives, but,

since there is always the possibility that a key may be compromised or that a representative will betray his organization, that trust may not be absolute.

It is clear that the Baldwin-Gramlich protocol can not be used in such a situation, since it relies on a third party available to all users at all times. Also, since the exchange of secrets involves immediate rather than eventual communication, users may find the Baldwin-Gramlich protocol, which involves eighteen separate messages, too lengthy to be practical. In this paper we will describe a protocol that also relies on a third party (the participating organizations), but in which that third party has to be available to a user only when he signs up for the system, not when he is actually testing for a match.

## 2. The Protocol

Our protocol is based on the Diffie-Hellman[2] protocol for secure key exchange. Like the Diffie-Hellman protocol, it uses the facts that $(X^Y)^Z = X^{YZ} = X^{ZY} = (X^Z)^Y$, where all exponentiations are taken over a finite field or $\mathbf{Z}_N$.

The basic idea is as follows: Suppose that A possesses secret $S_A$ and B possesses secret $S_B$, and that both share a common prime P. Suppose also that $S_A$ and $S_B$ are generators of the multiplicative group of $\mathbf{Z}_P$. Let A choose a secret number $M_A$, and let B choose a secret number $M_B$. A and B can match their secrets in the following manner:

[1] $A \rightarrow B : S_A^{M_A}$

[2] $B \rightarrow A : S_B^{M_B}$

[3] $A \rightarrow B : S_B^{M_B M_A}$

[4] $B \rightarrow A : S_A^{M_A M_B}$

where "A → B : M" means "A sends message M to B", and all exponentiations are modulo P.

This protocol clearly assumes a high degree of trust between A and B, since B can falsify a match with A by sending A back the message he received in the third step of the protocol. B (or A) can also find out if he has a match without revealing it to A by sending garbage in the last step of the protocol.

Another option is to give A the triple $[A, f_A(S_A), f_A]$ and B the triple $[B, f_B(S_B), f_B]$, where $f_A$ and $f_B$ are one-way functions (functions which are easy to compute but difficult to invert), and where each of these triples is cryptographically signed using private keys from public-private key pairs belonging to the participating organizations. A and B can then exchange their triples and calculate $f_A(S_B)$ and $f_B(S_A)$ without being able to find $S_A$ or $S_B$ directly. However, such a protocol allows a user who obtains another user's triple illicitly (say, for example, he is given it by a third user who checked for a match legitimately) without revealing to the other user whether or not there is a match. This could be a problem if someone steals a user's secret and wants to see who has a matching secret without tipping his hand. Thus such a protocol can not be considered desirable.

Our protocol will be an enhanced version of the Diffie-Hellman style protocol described above. We will rely on digital signatures of the users' organization to keep A and B honest in the first two steps. We will also describe a method of keeping A and B honest in the last two steps, based on the difficulty of factoring large integers.

We begin by having all the organizations in the system choose a common integer $N = (2pq + 1)(2r + 1)$, where p, q, and r all primes, and $2pq + 1$ and $2r + 1$ are primes. Thus $\phi(N)$, the order of the multiplicative group of $\mathbf{Z}_N$, is $4pqr$. The integer p will be made known to the users exchanging secrets, but the rest of $\phi(N)$ will remain secret. They also choose a common generator X of the multiplicative group of $\mathbf{Z}_N$. Next, the organizations choose a common integer u such that $u^p \equiv 1$ modulo N. This can be done by choosing a generator v of the multiplicative group of $\mathbf{Z}_N$ and letting $u = v^{4qr}$. Once this is done, the knowledge of N's factorization is not used again. This adds to the security of the protocol, since the information about N's factorization, whose secrecy our protocol relies on, can now be destroyed.

The organizations then encrypt all possible secrets by representing each secret as an integer a less than p, and computing $u^a$ modulo N. Thus any encrypted secret S has the property that $S^p \equiv 1$ modulo N. Each organization also chooses its own pair or pairs of public and private keys.

When User A belonging to ORG(A) signs up for the system, he receives N, p, an encrypted secret $S_A$, and the public keys of all organizations belonging to the system. He also receives integers $M_A$ and $D_A$ and a tuple $[A, ORG(A), S_A^{M_A}, X^{M_A}, (S_A^{M_A} \cdot X^{M_A})^{D_A}]$, where all exponentiations and multiplications are taken modulo N, and the tuple is signed with ORG(A)'s public-key signature. The tuple will be used to match A's secrets with those of other users, while A will keep the integers $M_A$ and $D_A$ secret.

Suppose that A and B wish to match secrets. The protocol runs as follows.

[1] $A \rightarrow B$: $[A, ORG(A), S_A^{M_A}, X^{M_A}, (S_A^{M_A} \cdot X^{M_A})^{D_A}]$, $sig_{ORG(A)}([A, ORG(A), S_A^{M_A}, X^{M_A}, (S_A^{M_A} \cdot X^{M_A})^{D_A}])$

B checks ORG(A)'s signature and computes $X^{M_A \cdot M_B}$.

[2] $B \rightarrow A$: $[B, ORG(B), S_B^{M_B}, X^{M_B}, (S_B^{M_B} \cdot X^{M_B})^{D_A}]$, $sig_{ORG(B)}([B, ORG(B), S_B^{M_B}, X^{M_B}, (S_B^{M_B} \cdot X^{M_B})^{D_B}])$

A checks ORG(B)'s signature and computes $X^{M_B M_A}$. The common value $K = X^{M_B M_A}$ will be used from now on as a conventional key, encrypting all messages between A and B (or rather the first m bits of $X^{M_B M_A}$, since $X^{M_B M_A}$ will be too large for a conventional key).

[3] $A \rightarrow B : E_K(OK,A)$

[4] $B \rightarrow A : E_K(OK,B)$

[5] $A \rightarrow B : E_K( [S_B^{M_B M_A}, (S_B^{M_B} \cdot X^{M_B})^{D_B M_A}])$

B multiplies $S_B^{M_B \cdot M_A}$ and $X^{M_B M_A}$ together, raises their product to the $D_B$, and checks that it is equal to the second item in the message received from A. He also checks that $(S_B^{M_B \cdot M_A})^p \equiv 1$ modulo N. He then checks whether or not $S_A^{M_A \cdot M_B} = S_B^{M_B M_A}$. If they are equal, he knows he has a match.

[6] $B \rightarrow A : E_K( [S_A^{M_A M_B}, (S_A^{M_A} \cdot X^{M_A})^{D_A M_B}])$

A multiplies $S_A^{M_A M_B}$ and $X^{M_A \cdot M_B}$ together, raises their product to the $D_A$, and checks that it is equal to the second item in the message received from B. He also checks that $(S_A^{M_A \cdot M_B})^p \equiv 1$ modulo N. He then checks whether or not $S_A^{M_A M_B} = S_B^{M_B M_A}$. If they are equal, he knows he has a match.

We first claim that A cannot find out B's encrypted secret. A knows $S_A^{M_A M_B}$ and therefore knows $S_A^{M_B}$, since he can find $M_A^{-1}$ modulo p. But in order to find $M_B$, even when he knows $S_A$, he would have to be able to find finite logarithms over $\mathbf{Z}_N$ with respect to the base $S_A$, where $S_A$ is a generator of the subgroup of order p of the multiplicative group of $\mathbf{Z}_N$. This problem can be made intractable if p is made large enough, even if the factorization of N becomes known.

Secondly, we present an heuristic argument to the effect that A cannot fake a match with B or send B garbage without his detecting it. Suppose that in the fifth step of the protocol A sends B a fake pair [W,Z]. In order for B not to detect the fraud, A would have to ensure both that $W^p \equiv 1$ modulo N and that $Z = W^{D_B} \cdot X^{M_B \cdot M_A D_B}$. A knows many possible values of W and Z satisfying the second equation, namely, $W = S_B^{M_B \cdot M_A \cdot C} \cdot X^{M_B \cdot M_A (C-1)}$ and $Z = (S_B^{M_B} \cdot X^{M_B})^{M_A \cdot C}$ where C is any integer. But he must also ensure that $W^p = (S_B^{M_B \cdot M_A \cdot C} \cdot X^{M_B \cdot M_A (C-1)})^p \equiv 1$ modulo N. Since $S_B^{M_B M_A \cdot C \cdot p} \equiv 1$ modulo N, A must choose C so that $X^{M_B M_A \cdot (C-1) \cdot p} \equiv 1$ modulo N. This he can only do by choosing an integer C such that C-1 is divisible by 4qr. But if A were knew 4qr, he would know $\phi(N) = 4pqr$, since he knows p. But if A knew $\phi(N)$, he wold be able to break any RSA (Rivest - Shamir - Adleman)[3] cryptosystem based on N, since given any integer e, he would be able to compute an integer $d = e^{-1}$ modulo $\phi(N)$ such that $Y^{e\,d} \equiv Y$ modulo N for any integer Y less than N. Nor does it seem likely that he would be able to find integers W and Z satisfying the appropriate equations without knowing 4qr. Thus his only choice is to let C = 1, and to send B $S_B^{M_B \cdot M_A}$.

Of course, if an organization's private key and some encrypted secrets become compromised, a user can easily fake matches or find out whether or not he has a match with another user without the other user knowing whether there is a match or not. Similarly, if $\phi(N)$ is compromised, a user can find out whether or not he has a match with another user without the other user finding out whether there is a match or not. We can detect fake matches by using a technique similar to that used by Baldwin and Gramlich. That is, we can give each user a set of fake tuples encrypting fake secrets which are random numbers. These fake tuples can be either be submitted along with the genuine tuples whenever a request for a match is received or initiated, or can be used to initiate fake requests for matches. In either case, if a fake tuple is matched, we can detect the fraud. However, the fake tuples are no protection against a user who merely wants to see who else matches his secret without letting that person know whether or not there is a match.

## 3. Conclusion and Open Problems

We have constructed a protocol by which users can verify each other's credentials in the absence of a continuously available third party without revealing their credentials to each other unless there is a match. However, the security of this protocol depends upon the intractability of two problems. The first is the problem

of finding logarithms to the base S over $\mathbf{Z}_N$ where $S^p \equiv 1$ modulo N. The second is the problem of factoring a large integer N when a large factor of $\phi(N)$ is known. Since the efficiency of the protocol increases as the size of N decreases, it is obviously desirable to obtain as good an estimate as possible of what size N and p must be for the above two problems to be intractable.

**References**


1. R. W. Baldwin and W. C. Gramlich, "Cryptographic Protocol for Trustable Match Making," *Proceedings of the 1985 Symposium on Security and Privacy*, pp. 92-100, IEEE Computer Society, Oakland, California, April 22-25, 1985.
2. W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. on Info. Theory*, vol. IT-22(6), pp. 644-654, Nov. 1976.
3. R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Exchanging Digital Signature and Public-Key Cryptosystems," *Comm. ACM*, vol. 21(2), pp. 120-126, Feb. 1978.