J. Cryptology (2002) 15: 139–148
DOI: 10.1007/s00145-001-0017-4

Journal of
CRYPTOLOGY

Authenticated Key Exchange Provably Secure against the Man-in-the-Middle Attack

Anna M. Johnston
Sandia National Laboratories,*
Albuquerque, NM 87185-1110, U.S.A.
ajohnst@sandia.gov

Peter S. Gemmell
Computer Science Department,
University of New Mexico,
Albuquerque, NM 87131-1386, U.S.A.
gemmell@cs.unm.edu

Communicated by Ernie Brickell

Received March 2000 and revised August 2001
Online publication 23 November 2001

Abstract. The standard Diffie–Hellman key exchange is suseptible to an attack known as the man-in-the-middle attack. Lack of authentication in the protocol makes this attack possible. Adding separate authentication to the protocol solves the problem but adds extra transmission and computation costs. Protocols which combine the authentication with the key exchange (an authenticated key exchange) are more efficient but until now none were provably secure against the man-in-the-middle attack. This paper describes an authenticated key exchange based on the difficulty of the qth-root problem, a problem believed to be equivalent to the discrete logarithm problem over groups of order q^2 (where q is a large prime) and parallel to the square-root problem over the ring modulo N, where N is a strong two prime composite integer. We show that mounting a man-in-the-middle attack for our protocol is equivalent to breaking the Diffie–Hellman problem in the group.

Key words. qth Roots, Discrete logarithms, Key exchange, Signature scheme.

1. Introduction

Astandard digital key exchange allows two parties to exchange publicly viewable data electronically, and with that data create a secret key known only by the two parties. The Diffie–Hellman key exchange does exactly this, and in general is very secure (see [4]). However, the Diffie–Hellman key exchange was not designed with any authentication. It

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

is suseptible to the man-in-the-middle or impersonator attack. In this attack an adversary impersonates one of the parties. Without authentication there is no way to distinguish a friend from a foe.

Two basic methods have been used to solve this key exchange authentication problem. The first method simply adds an authentication scheme to the exchange. Public key exchange data is not accepted unless it is accompanied by a valid digital signature of the public data. Although this solves the authentication problem it adds extra transmission, computation, and data management costs to the protocol.

The second method is to modify the Diffie–Hellman key exchange such that the digital authentication is built into the exchange (see [7] and [3]). Authentication is most often added to the key exchange by incorporating a good one-way function into the key exchange. The one-way functions most often used are standard hash functions. These functions are complicated and often hard to analyze. To date, none of these authenticated key exchanges has been provably secure against the man-in-the-middle attack.

The authentication property of our key exchange is based on a relatively new hard problem. The qth-root problem (described below) is believed to be equivalent to the discrete logarithm problem over groups of order q^2, where q is a large prime integer. This hard problem gives us a very simple one-way function: raising elements of order q^2 in the group to the qth power is a one-way function. The algebraic simplicity of this one-way function is what enables us to prove the security of the key exchange.

What we propose to do in this paper is to describe an authenticated key exchange provably secure against the man-in-the-middle attack. We will show that if the man-in-the-middle attack can be mounted against our key exchange, then we can solve the Diffie–Hellman problem.

2. The qth Root Problem

The qth root problem (see [1] and [6]) over a group G of order q^2 is parallel, in many ways, to the square-root problem modulo a two prime composite. Factoring the modulus is equivalent to the ability to find square roots, and the ability to find square roots enables factoring. Likewise, solving the discrete logs problem in G is equivalent to the ability to find qth roots in G, and the ability to find qth roots enables the solving of the discrete logarithm problem (see [2]). These problems convert the hard problems of factoring and discrete logs to root-finding problems.

A few characteristics of the problems disrupt their parallelism. These characteristic differences all stem from the fact that the square-root problem is tied to the factoring problem and the qth-root problem is tied to the discrete log problem over G. Each give good and bad cryptographic traits to its particular “root” problem.

The first major difference is the choice of group. An important cryptographic implementation detail is what group will the system be working over. The choice effects run time, security, power consumption, and the amount of data necessary for transmission. For example, modulo N arithmetic, where N is a large composite integer, is complicated and time consuming. On the other hand, if the group G was chosen such that the base operations are over $GF(2)$, much less time and effort are usually required. The square-root problem is tied to factoring, which limits the choice of group. The qth-root problem is tied to the discrete logarithm problem, allowing for many choices for the group.

The second difference is the possible existence of a trapdoor. This is both a plus and minus for both groups, depending on the cryptographic requirements. When the security of a system is based on factoring, knowledge of the factors is a trapdoor. This single piece of information allows the holder to compute square roots at will. No such trapdoor exists with the qth-root problem.[1] A trapdoor makes public key encryption and signature with message recovery possible. Without the trapdoor, designing public key encryption and signature with message recovery is difficult if not impossible. On the other hand, if only a signature is needed, then the fact that such a trapdoor does not exist lends credence to the system, especially in situations with mutually mistrusting parties.

This paper describes a simple authenticated key exchange based on the qth-root problem.

3. Why Is an Authenticated Key Exchange Necessary?

In a standard Diffie–Hellman key exchange (see Fig. 1), a shared secret is established by exchanging a common base element raised to a one-time random, secret value. The users create the shared secret by raising the value received by the random secret they have created. Because the exponentiation is commutative, these values will be the same for both users. However, because the discrete logarithm problem is so hard, no one other than these two users should be able to create this shared secret

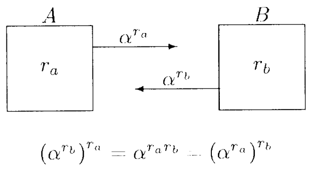
The two users can now use the shared secret as a key for a symmetric cryptosystem.

This technique is simple, fairly efficient, and secure. Computing the secret key from the two public portions of the key is called the Diffie–Hellman problem and it appears to be as difficult as the discrete logarithm problem.

Definition 1 (The Diffie–Hellman Problem). Given

- G: A cyclic group (or subgroup) for which discrete logarithms are "hard".
- α: A primitive element in G.
- (α^a, α^b): Public portions of a key exchange.

Find the secret key α^{ab}. Although never proven, this problem is considered to be equivalent to the discrete logarithm problem.



A
B
r_a
r_b
α^{r_a}
α^{r_b}
$(\alpha^{r_b})^{r_a} = \alpha^{r_a r_b} = (\alpha^{r_a})^{r_b}$

Fig. 1. Standard Diffie–Hellman key exchange.

[1] The discrete logarithm problem is intractable and without trapdoors for most large groups with non-smooth order. However, there are some poor choices of groups G for which the discrete logarithm problem is subexponential—see [8] and [5].

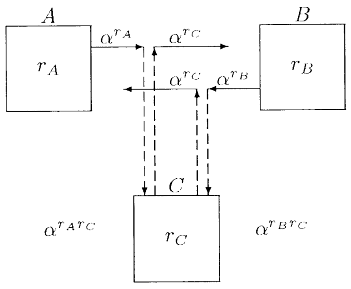


Fig. 2. Impersonation attack.

There is only one major problem with the scheme: it is not authenticated. This means that user A has no way of knowing if the α^{r_B} actually came from user B. A third party (C) could have intercepted the transmission from B and substituted their own value (say α^{r_C}). User A believes that the key generated is shared with user B, when in fact it is shared with user C. Any transmissions sent by user A to B can be intercepted by user C and read. User C can also send messages to user A, posing as user B. If user C intercepts and replaces transmissions from both users (A and B), user C can effectively tap an encrypted line. Neither user would be able to tell that any thing is wrong with their system and both would believe that the messages they were sending could only be read by the other user. This attack is called the impersonation or man-in-the-middle attack (see Fig. 2).

Definition 2 (Generalized Impersonation Attack). Let E be a key exchange protocol with public long term parameters P_a, P_b, one-time public parameters D_a, D_b, which returns a key K: $E\,(P_a, P_b, D_a, D_b) = K$. An impersonation attack on E computes a one-time public D' and a matching key K' given D_a, P_a, P_b: $E\left(P_a, P_b, D_a, D'\right) = K'$. The attack can be reduced to finding the function F, where

$$F(P_a, D_a, P_b) = [D', K'].$$

4. Authenticated Key Exchange

The main purpose of an authenticated key exchange is to prevent the impersonation attack. An authenticated key exchange does not guarantee that the two parties share the same key. The two parties will share the same key only if they receive uncorrupted data from each other. An authenticated key exchange does guarantee that the two parties either

(1) share a common key or
(2) share no key with each other *OR* anyone else.

The impersonation attack can be thwarted in several ways. The first technique adds an authentication mechanism into the key exchange: add a digital signature of the public version of the key half. This prevents the impersonation attack at the cost of a digital signature.

A less costly solution is to use a key exchange with authentication built in: an authenticated key exchange protocol. The following scheme differs from the basic Diffie–Hellman scheme by applying a one-way function to the random value and adding a long term secret key to the scheme. In this simplified description it is assumed that each user has an authenticated public version of the other user's secret key and that the secret keys remain secret. It is described in [7] and what follows is a generalized overview:

Components

- G: A group for which discrete logarithms are "hard".
- α: An element whose order is divisible by a large prime divisor.
- x_i: Long term secret integer (modulo the order of α) for user i.
- γ_i: The public version of x_i, namely $\gamma_i \equiv \alpha^{x_i}$.
- r_i: A one-time random secret integer (modulo the order of α) for user i.
- μ_i: The public version of r_i, namely $\mu_i \equiv \alpha^{r_i}$.
- H: This function converts an element in G into an integer modulo the order of α.

In standard Diffie–Hellman key exchange the μ_i values are exchanged and combined with the user's one-time random secret value to obtain the final key. This scheme exchanges the same one-time public values (the μ_i values) but combines them with the long term keys in such a way that the impersonation attack is blocked.

Authenticated Key Exchange (see Fig. 3)

1. Transmit the μ_i values.
2. Receive μ_j.
3. Compute $\theta_i \equiv \mu_j \gamma_j^{H(\mu_j)} \equiv \alpha^{x_j H(\mu_j)+r_j}$.
4. Compute the shared key: $\theta_i^{x_i H(\mu_i)+r_i} \equiv \alpha^{(x_i H(\mu_i)+r_i)(x_j H(\mu_j)+r_j)}$.

Whilethe long term secret is necessary for authentication, without the one-way function on the random data (i.e., using μ_i in the exponent) the impersonation attack would still be viable. Here is how it works. In the description below C will impersonate user A to user B:

1. C (the man-in-the-middle) has access to γ_A and prevents the true μ_A from reaching B.
2. C generates a random value t.
3. Let t equal $x_A + r'_A$. Neither x_A nor r'_A is known.

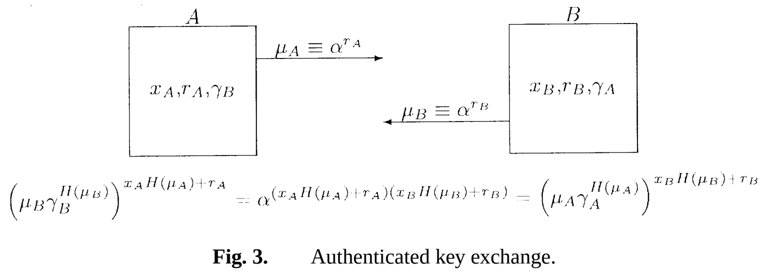

A
B
x_A, r_A, γ_B
$\mu_A \equiv \alpha^{r_A}$
$\mu_B \equiv \alpha^{r_B}$
x_B, r_B, γ_A
$\left(\mu_B \gamma_B^{H(\mu_B)}\right)^{x_A H(\mu_A)+r_A} = \alpha^{(x_A H(\mu_A)+r_A)(x_B H(\mu_B)+r_B)} = \left(\mu_A \gamma_A^{H(\mu_A)}\right)^{x_B H(\mu_B)+r_B}$

Fig. 3. Authenticated key exchange.

4. C now generates $\mu'_A \equiv \alpha^{r'_A}$ by computing $\mu'_A \equiv \alpha^t \gamma_A^{-1}$.
5. The shared key will be $\alpha^{t(x_B + r_B)}$, which C creates with knowledge of t.

4.1. *How the Attack is Blocked*

In the previous authenticated key exchange the one-way function, g, on the random public component was simple exponentiation: μ_i is used in the exponent to create the key. An imposter would still be able to generate a random t value, but removing the public version of the long term private key and solving for the random one-time public key appears to be a very hard problem.

Given γ_A, γ_B, and μ_B the imposter must generate a matching pair μ_C, K, where

$$K = \alpha^{(x_A H(\mu_C) + r_C)(x_B H(\mu_B) + r_B)},$$

where r_C is the discrete logarithm, base α, of μ_C. The imposter can create $\alpha^{(x_B H(\mu_B) + r_B)}$ which reduces the problem to finding a pair

$$\mu_C, [t = x_A H(\mu_C) + r_C].$$

Choosing a random r_C and solving for t involves solving a discrete log problem. Choosing a random t and solving for μ_C seems more tractable. Start by generating a random $t' = x_A + r_c H(\mu_C)^{-1}$. This allows us to strip off the x_A in the exponent and group all the unknowns together. If μ_C can be found, then the original t value is easily derived ($t = t' H(\mu_C)$).

With this data the value $\mu_C^{H(\mu_C)^{-1}}$ can be computed. However, computing μ_C from this value appears to be as difficult as the discrete log problem.

5. Simplified Authenticated Key Exchange

In order to simplify the key exchange the first question to ask is what purpose does the one-way function serve? Its main purpose is to prevent an imposter from using the public information of two legitimate users to create a key and a public value. The imposter wants the key and public value to have the property that if the public value is sent to one of the legitimate users, they will use it, along with their private key and the other legitimate user's public key, to create a key which the imposter can duplicate. This is what the second version of the impersonation attack does (see Fig. 4).

The scheme presented in this paper also uses a one-way function to thwart an impersonation attack. The one-way function is much simpler (arithmetically) however, and thus more can be proved about it. In [2] it was shown that finding qth roots in a group of order q^2 was as equivalent to finding discrete logarithms in a subgroup of order q. Raising elements to the qth power in a group of order q^2 is a simple one-way function based on the discrete logarithm problem, and the one we use for this simplified authenticated key exchange (see Fig. 5). The scheme is the following:

Components

- G: A group (or subgroup) for which discrete logarithms are "hard" and which has order q^2.

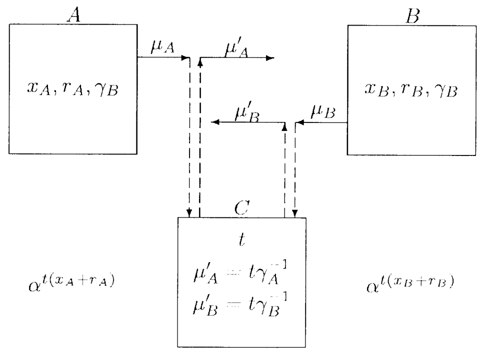


Fig. 4. Impersonation attack—version 2.

- α: An element whose order is q^2.
- x_i: Long term secret integer (modulo q^2) for user i.
- γ_i: The public version of x_i, namely $\gamma_i \equiv \alpha^{x_i}$.
- r_i: A one-time random secret integer (modulo q^2) for user i.
- μ_i: The public version of r_i, namely $\mu_i \equiv \alpha^{r_i}$.

Simplified Authenticated Key Exchange (SAKE)

1. Transmit the μ_i values.
2. Receive μ_j.
3. Compute $\theta_j \equiv \mu_j^q \gamma_j \equiv \alpha^{x_j+qr_j}$.
4. Compute the shared key: $\theta_j^{x_i+qr_i} \equiv \alpha^{(x_i+qr_i)(x_j+r_jq)}$.

The simplicity of the one-way function has several benefits. First, it allows us to show that performing an impersonator attack on SAKE is equivalent to solving the Diffie–Hellman problem. Second, it may allow for efficiency improvements in hardware as the one-way function and exponent, q, can be chosen or designed as needed. One draw back to this scheme is that it requires a group (or subgroup) of order q^2. Finding such a group is slightly more difficult (though this is one time work) and for some groups, such as elliptic curves, it could require more transmission bits.

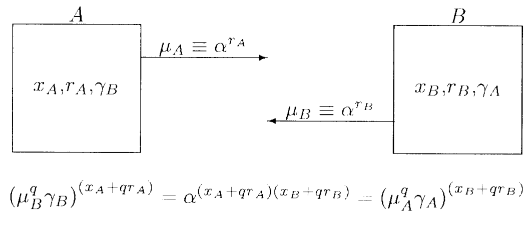

A
x_A, r_A, γ_B
$\mu_A \equiv \alpha^{r_A}$
B
x_B, r_B, γ_A
$\mu_B \equiv \alpha^{r_B}$
$(\mu_B^q \gamma_B)^{(x_A+qr_A)} = \alpha^{(x_A+qr_A)(x_B+qr_B)} = (\mu_A^q \gamma_A)^{(x_B+qr_B)}$

Fig. 5. Simplified authenticated key exchange.

6. Mapping to the Diffie–Hellman Problem

Although standard Diffie–Hellman key exchange is vulnerable to the impersonation attack, the fundamental Diffie–Hellman problem appears to be hard. This section shows that performing an impersonation attack against SAKE would break the Diffie–Hellman problem.

We have:

- G: A group (or subgroup) for which discrete logarithms are "hard" and which has order q^2.
- α: An element whose order is q^2.

An impersonation attack against SAKE (see Definition 2) implies that the attacker, knowing only α^X, α^Y, and α^Z, is able to generate the pair

$$[\alpha^R, K = \alpha^{(X+qZ)(Y+qR)}].$$

With this matched pair, an impersonator would send α^R to the user with the secret key X. This user would think that a key had been exchanged with the user with the secret key Y. Instead the key would be shared with the impersonator. This section shows that implementing the impersonation attack is equivalent to breaking the Diffie–Hellman problem.

6.1. *The ORACLE*

The ORACLE is a function which takes three inputed values and returns two. In particular the ORACLE is defined by the function f:

$$f(\alpha^X, \alpha^Y, \alpha^Z) = [\alpha^R, K = \alpha^{(X+qZ)(Y+qR)}].$$

6.2. *The Reduction*: *Solving the Diffie–Hellman Problem Using the ORACLE*

Given the ORACLE f, solving the Diffie–Hellman problem requires three calls to the oracle, five exponential group operations, and standard group operations.

To Compute α^{XY}

1. Choose three random values in $\mathcal{Z}^*_{q^2}$, Z_1, Z_2, Z_3.
2. Call the oracle on inputs α^X, α^Y, and α^{Z_1}:

$$f(\alpha^X, \alpha^Y, \alpha^{Z_1}) = [\alpha^{R_1}, K_1 = \alpha^{(X+qZ_1)(Y+qR_1)} = \alpha^{XY+q(XR_1+YZ_1)}].$$

3. Call the ORACLE on inputs α^X, α^{R_1}, and α^{Z_2}:

$$f(\alpha^X, \alpha^{R_1}, \alpha^{Z_2}) = [\alpha^{R_2}, K_2 = \alpha^{(X+qZ_2)(R_1+qR_2)} = \alpha^{XR_1+q(XR_2+Z_2R_1)}].$$

4. Call the ORACLE on inputs α^Y, α^{Z_1}, and α^{Z_3}:

$$f(\alpha^Y, \alpha^{Z_1}, \alpha^{Z_3}) = [\alpha^{R_3}, K_3 = \alpha^{(Y+qZ_3)(Z_1+qR_3)} = \alpha^{YZ_1+q(YR_3+Z_3Z_1)}].$$

5. Raise K_2 and K_3 to the $-q$ power to obtain

$$K_2^{-q} = \alpha^{-qXR_1},$$
$$K_3^{-q} = \alpha^{-qYZ_1}.$$

6. Multiply K_1, K_2^{-q}, and K_3^{-q} to obtain the Diffie–Hellman value:

$$K_1K_2^{-q}K_3^{-q} = \alpha^{XY+q(XR_1+YZ_1)-q(XR_1+YZ_1)} = \alpha^{XY}.$$

6.3. *Conclusions of the Proof*

With the imposter ORACLE and the public versions of the key, the Diffie–Hellman problem can be easily broken. This shows breaking SAKE with an imposter attack is as difficult as breaking the Diffie–Hellman problem.

7. Conclusions

The SAKE algorithm provides a simple, authenticated method for exchanging keys which is provably secure against an imposter attack. The algorithm is flexible: it works over any group for which discrete logarithms are difficult and the square of a large prime divides the order. For example, various elliptic curve or F_P are suitable.

Another benefit is that the prime q is fixed, allowing for easy optimization of the implementation. The group can even be created with a particular prime q in mind. A prime with low bit density will improve the efficency.

References

[1] E. Bach and J. Shallit, *Algorithmic Number Theory - Volume* 1: *Efficient Algorithms*, second edn., MIT Press, Cambridge, MA, 1997.

[2] C. Beaver, P. Gemmell, A. Johnston, and W. Newmann, On the cryptographic value of the qth-root problem, in *Proceedings of the International Conference on Information and Computer Security*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1999, pp. 135–142.

[3] S. M. Bellovin and M. Merritt, Encrypted key exchange: password-based protocols secure against dictionary attacks, in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, May 1992, pp. 72–84.

[4] W. Diffie and M. Hellman, New directions in cryptography, in *IEEE Transactions on Information Theory*, vol. 22 (November 1976), pp. 644–654.

[5] G. Frey and H.-G. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves, *Mathematics of Computation*, vol. 62 (1994), pp. 865–874.

[6] A. Johnston, A generalized qth-root algorithm, in *Proceedings of the Tenth Annual ACM–SIAM Symposium on Discrete Algorithms*, January 1999.

[7] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, An Efficient Protocol for Authenticated Key Agreement, Technical Report CORR 98-05, Department of C&O, University of Waterloo, Ontario, March 1998.

[8] A. Menezes, T. Okamoto, and S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, in *Proceedings of the* 23*rd Annual ACM Symposium on Theory of Computing*, 1991, pp. 80–89.

[9] M. O. Rabin, Digitalized Signatures and Public Key Functions as Intractable as Factorization. Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Cambridge, MA, 1979.

[10] K. H. Rosen, *Elementary Number Theory and its Applications*, Addison-Wesley, Reading, MA, third edn., 1993.

[11] D. Shanks, Five number-theoretic algorithms, in *Proceedings of the Second Manitoba Conference on Numerical Mathematics*, no. VII, 1972, pp. 51–70.