VOL. 7, NO. 2, FEBRUARY 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

SAFETY MEASURES AGAINST MAN-IN-THE-MIDDLE ATTACK IN KEY EXCHANGE

C. Krishna Kumar[1], G. Jai Arul Jose[1], C. Sajeev[1] and C. Suyambulingom[2]
[1]Sathyabama University, Chennai, India
[2]Department of Mathematics, TAU, Coimbatore, India
E-Mail: kkumarmalar@yahoo.com

ABSTRACT

Several techniques have been proposed for the distribution of public keys. The ability to distribute cryptographic keys securely has been a challenge for centuries. The Diffie-Hellman key exchange protocol was the first practical solution to the key exchange dilemma. The Diffie-Hellman protocol allows two parties to exchange a secret key over unsecured communication channels without meeting in advance. The secret key can then be used in a symmetric encryption application, and the two parties can communicate securely. However, if the key exchange takes place in certain mathematical environments, the exchange becomes vulnerable to a specific man-in-the-middle attack, first observed by Vanstone [1]. We explore this man-in-the-middle attack, analyze countermeasures against the attack.

Keywords: cryptography, public key, key exchange, man-in-the-middle attack.

1. INTRODUCTION

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [2] and is generally referred to as Diffie-Hellman key exchange. A number of commercial products employ this key exchange technique. The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values. The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

Briefly, we can define the discrete logarithm in the following way. First, we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to p - 1. That is, if a is a primitive root of the prime number p, then the numbers a mod p, a^2 mod p,..., a^{p-1} mod p are distinct and consist of the integers from 1 through p - 1 in some permutation. For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that $b \equiv a^i \pmod{p}$ where $0 \equiv i \leq (p - 1)$ the exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as $\mathrm{dlog}_{a,p}$ (b).

2. DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

There are two publicly known numbers: a prime number q and α an integer that is a primitive root of q. suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{XA}$ mod q. Similarly, user B independently selects a random integer $X_A < q$ and computes $Y_B = \alpha^{XB}$ mod q. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{XA}$ mod q and user B computes the key as $K = (Y_A)^{XB}$ mod q. These two calculations produce identical results:

$K = (Y_B)^{XA} \bmod q$
$= (\alpha^{XB} \bmod q)^{XA} \bmod q$
$= (\alpha^{XB})^{XA} \bmod q$
by the rules of modular arithmetic
$= (\alpha^{XB\,XA} \bmod q$
$= (\alpha^{XA})^{XB} \bmod q$
$= (\alpha^{XA} \bmod q)$
$= (\alpha^{XA} \bmod q)^{XB} \bmod q$
$= (Y_A)^{XB} \bmod q$

The result is that the two sides have exchanged a secret value. Furthermore, because X_A and X_B are private, an adversary only has the following ingredients to work with: q,α Y_A, and Y_B. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute
$X_B = \mathrm{dlog}_{\alpha,\,q}(Y_B)$

The adversary can then calculate the key K in the same manner as user B calculates it.
The example below illustrates the procedure.

a) Alice and Bob agree on q = 37 and □ = 2.
b) Alice chooses x = 14 and sends Bob 30 ($\equiv 2^{14}$ mod37). $A \rightarrow B$:30
c) Bob chooses y = 23 and sends Alice 5($\equiv 2^{23}$ mod37). $B \rightarrow A$:5
d) Bob receives 30 and computes 30^{23} mod 37 = 28
e) Alice receives 5 and computes 5^{14} mod 37 = 28

Alice and Bob have agreed upon 28 as their secret key

Obviously, a much larger value of *p* is required than used in the example to make the key agreement

VOL. 7, NO. 2, FEBRUARY 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

potentially secure. If the prime number 37 was used, Eve could simply try all possible values of $2^{x\ y}$ mod 37. Because 2 is a primitive root modulo 37, this can take 36 values. A key space with only 36 possibilities can be exhausted with ease. However, if the prime number used is large enough, no computing power available today can exhaust the key space. For instance, most applications recommend 1024-bit primes [3]. This correlates to a number of about 300 digits and makes searching the key space one by one infeasible.

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

The key exchange protocols

Figure-1 shows a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A, calculate Y_A, and send that to user B. User B responds by generating a private value X_B calculating Y_B, and sending Y_B to user A. Both users can now calculate the key. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message.

As an example of another use of the Diffie-Hellman algorithm, suppose that a group of users (e.g., all users on a LAN) each generate a long-lasting private value X_i (for user i) and calculate a public value Y_i. These public values, together with global public values for q and a, are stored in some central directory. At any time, user j can access user i's public value, calculate a secret key, and use that to send an encrypted message to user A. If the central directory is trusted, then this form of communication provides both confidentiality and a degree of authentication. Because only i and j can determine the key, no other user can read the message (confidentiality). Recipient i knows that only user j could have created a message using this key (authentication). However, the technique does not protect against replay attacks.

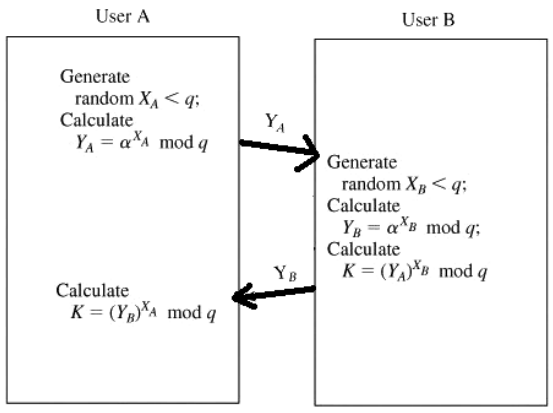
User A
User B
Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$
Y_A
Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$
Y_B
Calculate $K = (Y_B)^{X_A} \bmod q$

Figure-1. The key exchange protocols.

3. THE MAN-IN-THE-MIDDLE ATTACK

Wiener and van Oorschot [3] noted that, if certain primes are used, a potentially fatal protocol attack on the Diffie-Hellman key exchange protocol becomes possible. The idea is based on forcing the parties to agree on a shared key that resides in a subgroup of the cyclic group Zq. If the order of the subgroup is small enough, an adversary can exhaustively search the subgroup, retrieve the secret key, and eavesdrop on the communication of Alice and Bob.

The protocol depicted in Figure-1 is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

a) Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
b) Alice transmits Y_A to Bob.
c) Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates $K2 = (Y_A)^{X_{D2}}$ mod q.
d) Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{X_E}$ mod q.
e) Bob transmits X_A to Alice.
f) Darth intercepts X_A and transmits Y_{D2} to Alice. Darth calculates $K1 = (Y_B)^{X_{D1}}$ mod q.
g) Alice receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A}$ mod q.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

a) Alice sends an encrypted message M: E (K2, M).
b) Darth intercepts the encrypted message and decrypts it, to recover M.
c) Darth sends Bob E (K1, M) or E (K1, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures.

4. SAFETY MEASURES AGAINST THE ATTACK

To prevent from this attack, Alice and Bob have several options. The easiest method is to force authentication prior to the key exchange. The simple scheme is as follows:

(1) $X \rightarrow A$: $ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$

(2) $A \rightarrow Y$: $E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])$

X double encrypts a message M first with X's private key, PR_x and then with Y's public key, PU_y. This is a signed, secret version of the message. This signed message, together with X's identifier, is encrypted again with PR_x and, together with ID_X, is sent to A. The inner, double-encrypted message is secure from the arbiter (and

VOL. 7, NO. 2, FEBRUARY 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

everyone else except Y). However, A can decrypt the outer encryption to assure that the message must have come from X (because only X has PR_x). A checks to make sure that X's private/public key pair is still valid and, if so, verifies the message. Then A transmits a message to Y, encrypted with PR_a. The message includes ID_X, the double-encrypted message, and a timestamp.

This scheme has a number of advantages over the preceding two schemes. First, no information is shared among the parties before communication, preventing alliances to defraud. Second, no incorrectly dated message can be sent, even if PR_x is compromised, assuming that PR_a is not compromised. Finally, the content of the message from X to Y is secret from A and anyone else. However, this final scheme involves encryption of the message twice with a public-key algorithm.

Authentication

As an example we will discuss the STS (Station - to - Station Protocols). STS is a three-pass variation of the basic Diffie-Hellman protocol that allows the establishment of a shared secret key between two parties with mutual entity authentication and mutual explicit key authentication [1]. The STS employs digital signatures. A digital signature of a message is a number dependent on some secret known only to the signer; and, additionally, on the content of the message being signed [1]. The STS protocol is frequently employed with the RSA signature scheme. To employ an RSA signature scheme, public and private key pairs must first be generated. RSA signature scheme key generation steps [1]:

a) Generate two large distinct random primes p and q, each roughly the same size.
b) Compute $n = pq$ and $\phi = (p - 1)(q - 1)$
c) Select a random integer e,$1 < e < \phi$, such that $\gcd(e, \phi) = 1$
d) Use the extended Euclidean algorithm to compute the unique integer d,$1 < d < \phi$ such that $ed \equiv 1 \pmod{\phi}$
e) The user's public key is (n, e) and the user's private key is d

Now, if a user Alice wants to sign a message m, and a user Bob wants to verify the message signature, the remaining steps of the protocol must be completed.
RSA signature scheme protocol steps:

i) Signature generation

a) Compute $\underset{.}{m}'$ $R(m)$, an integer in the range [0, n-1]
b) Compute $s = \underset{.}{m}'d \bmod n$
c) Alice's signature for m is s.

ii) Signature verification

a) Obtain Alice's authentic public key (n, e)
b) Compute $\underset{.}{m} = s^e \bmod n$
c) Recover $m = R^{-1}$ *mod* $(\underset{.}{m}')$

With the knowledge of a digital signature scheme, in particular RSA, we can move onto the STS protocol. If we let E denote a symmetric encryption algorithm, and S_A (m) denote Alice's signature on m, the protocol is as follows:

1. Set up

a) A prime number p and generator α of Z_p^* ($2 \le p \le p - 2$) are selected and published.
b) Alice selects RSA public and private signature keys (n_A, e_A) d_A (Bob selects analogous keys). Assume each party has access to authentic copies of the other's public key.

2. Action

a) Alice generates a secret random x, $1 \le x \le p - 2$ and sends to Bob $\alpha^x \bmod p$.
$A \rightarrow B$: $\alpha^x \bmod p$ (message 1)
b) Bob generates a secret random y, $1 \le y \le p - 2$, and computes the shared key $k = (\alpha^x)^y \bmod p$. Bob signs the concatenation of both exponentials, encrypts this using the computed key, and sends to Alice.
$B \rightarrow A : a^y \bmod p, E_k(S_B(\alpha^y, \alpha^x))$ (message 2)
c) Alice computes the shared key $(\alpha^x)^y \bmod p$, decrypts the encrypted data, and uses Bob's public key to verify the received value as the signature on the hash of the cleartext exponential received and the exponential sent in message 1. Upon successful verification, Alice accepts that k is actually shared with Bob, and sends Bob an analogous message.
$A \rightarrow B : E_k(S_A(\alpha^x, \alpha^y))$ (message 3)
d) Bob similarly decrypts the received message and verifies Alice's signature therein. If successful, Bob accepts that k is actually shared with Alice.

The exchanged exponentials are digitally signed and retransmitted during the STS protocol. Therefore, Eve cannot alter the original exponentials without triggering a failure during Alice and Bob's key agreement. This precludes the man-in-the-middle attack we have focused on and defends Alice and Bob's key exchange against several other possible active man-in-the-middle attacks.

5. CONCLUSIONS

Possible future efforts include coding and implementing the man-in-the middle attack on active communications to test the theory laid out in this paper. It is possible that analyzing the given prime number, capturing the required messages, altering those messages, and forwarding the messages to the intended recipients will be too time-consuming.

REFERENCES

[1] P. C. van Oorschot and M. J. Wiener. 1996. On Diffie-Hellman Key Agreement with Short Exponents. EUROCRYPT'96, LNCS 1070, Springer-Verlag. pp. 332-343.

[2] Diffie W. and Hellman M. 1976. Multiuser Cryptographic Techniques. IEEE Transactions on Information Theory. November.

VOL. 7, NO. 2, FEBRUARY 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

[3] J. Menezes, P. C. van Oorshot and S. A Vanstone. 1997. Handbook of Applied Cryptography. CRC Press, New York, USA.

[4] Popek G. and Kline C. 1979. Encryption and Secure Computer Networks. ACM Computing Surveys. December.

[5] Kohnfelder L. 1978. Towards a Practical Public-Key Cryptosystem. Bachelor's Thesis, M.I.T. May.

[6] Denning D. 1983. Protecting Public Keys and Signature Keys. Computer. February.

[7] Merkle R. Secrecy. 1979. Authentication and Public Key Systems. Ph.D. Thesis. Stanford University, California, USA. June.

[8] S. Harris. 2005. All in One CISSP. McGraw-Hill, San Francisco, CA.

[9] T. Agoh. 2000. On Sophie Germain Primes. Tatra Mt. Math. Publ. p. 20.

[10]W. Trappe and L. Washington. 2006. Introduction to Cryptography with Coding Theory. 2nd Edition. Pearson, Upper Saddle River, NJ.

[11]M. Agrawal, N. Kayal and N. Saxena. 2004. PRIMES are in P. Annals of Mathematics. p. 160.

[12]K. H. Rosen. 2007. Discrete Mathematics and Its Applications. 6th Edition. McGraw Hill, San Francisco, CA.

[13]Rivest R. and Shamir A. 1984. How to Expose an Eavesdropper. Communications of the ACM. April.