ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

A Study on Asymmetric Key Exchange Authentication Protocols

Dr. D. S. R. Murthy, B. Madhuravani, G. Sumalatha

Abstract*– *Key exchange protocols enable the use of shared-key cryptography to protect transmitted data over insecure networks. The key exchange protocols are classified into i) symmetric ii) asymmetric (public key). Many public-key-based (asymmetric) key-exchange protocols already exist and have been implemented for a variety of applications and environments. We present a general framework for authentication problems and design principles of different asymmetric key exchange protocols. The analysis of these protocols also includes merits and demerits of each protocol. The study of these protocols emphasizes various observations that can motivate researchers in key management issues of networks.

***Index Terms*– Asymmetric Cryptography, Key Exchange, Shared–key, Symmetric Cryptography.**

I. INTRODUCTION

In communications security design, one of the most important questions is whether an opponent will ever have unsupervised access to the equipment. If the answer is no, then we can greatly simplify the design by storing long term secrets. However, the equipment will then have to be well guarded at all times. This may be feasible for military equipment, but in the commercial world, physical security procedures are generally insufficient to stop an opponent from getting occasional access. It follows that we must either use tamper resistant hardware, or avoid using long term secrets. Key establishment is the process by which two (or more) entities establish a shared secret key. The key may subsequently be used to achieve some crypto-graphic goal, such as confidentiality or data integrity. Ideally, the established key should have precisely the same attributes as a key established face-to-face for example, it should be shared by the (two) specified entities, it should be distributed uniformly at random from the key space, and no unauthorized entity should learn anything about the key. Key establishment protocols come in various flavors. In key transport protocols, a key is created by one entity and securely transmitted to the second entity, while in key agreement protocols both parties contribute information which is used to derive the shared secret key. In symmetric protocols the two entities a priori possess common secret information, while in asymmetric protocols the two entities share only public information that has been authenticated.

II. KEY EXCHANGE

A common cryptographic technique is to encrypt each individual conversation with a separate key. This is called a session key, because it is used for only one particular communications session [1]. Session keys are useful because they only exist for the duration of the communication. How this common session key gets into the hands of the conversant can be a complicated matter.

A. Key Exchange with Symmetric Cryptography

This protocol assumes that Alice and Bob, users on a network, each share a secret key with the Key Distribution Center (KDC).

(1) Alice calls Trent and requests a session key to communicate with Bob.
(2) Trent generates a random session key. He encrypts two copies of it: one in Alice's key and the other in Bob's key. Trent sends both copies to Alice.
(3) Alice decrypts her copy of the session key.
(4) Alice sends Bob his copy of the session key.
(5) Bob decrypts his copy of the session key.
(6) Both Alice and Bob use this session key to communicate securely.

This protocol relies on the absolute security of Trent, who is more likely to be a trusted computer program than a trusted individual. If Mallory corrupts Trent, the whole network is compromised. He has all of the secret keys that Trent shares with each of the users; he can read all past communications traffic that he has saved, and all future communications traffic. All he has to do is to tap the communications lines and listen to the encrypted message traffic. The other problem with this system is that Trent is a potential bottleneck. He has to be involved in every key exchange. If Trent fails, that disrupts the entire system.

B. Key Exchange with Public-Key Cryptography

Alice and Bob use public-key cryptography to agree on a session key, and use that session key to encrypt data. In some practical implementations, both Alice's and Bob's signed public keys will be available on a database. This makes the key-exchange protocol even easier, and Alice can send a secure message to Bob even if he has never heard of her:

(1) Alice gets Bob's public key from the KDC.
(2) Alice generates a random session key, encrypts it using Bob's public key, and sends it to Bob.
(3) Bob then decrypts Alice's message using his private key.
(4) Both of them encrypt their communications using the same session key.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

C. Man-in-the-Middle Attack

While Eve cannot do better than try to break the public-key algorithm or attempt a ciphertext-only attack on the ciphertext, Mallory is a lot more powerful than Eve. Not only can he listen to messages between Alice and Bob, he can also modify messages, delete messages, and generate totally new ones. Mallory can imitate Bob when talking to Alice and imitate Alice when talking to Bob [2]. Here's how the attack works:

(1) Alice sends Bob her public key. Mallory intercepts this key and sends Bob his own public key.
(2) Bob sends Alice his public key. Mallory intercepts this key and sends Alice his own public key.
(3) When Alice sends a message to Bob, encrypted in "Bob's" public key, Mallory intercepts it. Since the message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Bob's public key, and sends it on to Bob.
(4) When Bob sends a message to Alice, encrypted in "Alice's" public key, Mallory intercepts it. Since the message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Alice's public key, and sends it on to Alice. Even if Alice's and Bob's public keys are stored on a database, this attack will work. Mallory can intercept Alice's database inquiry and substitute his own public key for Bob's. He can do the same to Bob and substitute his own public key for Alice's. Or better yet, he can break into the database surreptitiously and substitute his key for both Alice's and Bob's. Then he simply waits for Alice and Bob to talk with each other, intercepts and modifies the messages, and he has succeeded.

This man-in-the-middle attack works because Alice and Bob have no way to verify that they are talking to each other. Assuming Mallory doesn't cause any noticeable network delays, the two of them have no idea that someone sitting between them is reading all of their supposedly Secret Communications.

IV. KEY EXCHANGE AUTHENTICATION PROTOCOL

A. Diffie-Hellman Session Key Agreement

One of the difficulties in the application of the symmetric key (or private key as is generally known) in classical cryptography is key management ElGamal (1985 and Abdalla *et al*., 1985). Key management is the process of distributing keys to the rightful owners without the man-in-middle intercepting them. In Diffie-Hellman algorithm [3], this dilemma of key distribution is solved. Diffie-Hellman key exchange forms the basis of the modern cryptosystem known as the public-key exchange [4]. A secret (session) key is only created when needed. There is no need to store keys for a long period of time which exposes them to increased vulnerability. This key exchange does not require pre-existing infrastructure. It can easily be extended to 3 or more entities.

The math is simple.

- First, Alice and Bob agree on a large prime, n and g, such that g is primitive mod n.
- These two integers don't have to be secret.
- Alice and Bob can agree to them over some insecure channel.
- They can even be common among a group of users. It doesn't matter.

Then, the protocol goes as follows:

(1) Alice chooses a random large integer x and sends Bob
$X = g^x \bmod n$
(2) Bob chooses a random large integer y and sends Alice
$Y = g^y \bmod n$
(3) Alice computes
$k = Y^x \bmod n$
(4) Bob computes
$k' = X^y \bmod n$
Both k and k' are equal to $g^{xy} \bmod n$.

No one, listening on the channel, can compute that value; they only know n, g, X, and Y. Unless they can compute the discrete logarithm and recover x or y, they do not solve the problem. So, k is the secret key that both Alice and Bob computed independently. This algorithm depends on a mathematical problem: the discrete logarithm problem for its security. If and when this problem is solved, this protocol will be useless. The choice of prime numbers has a substantial impact on the system. It is computationally intensive. As a result it is subject to clogging attack [5] in where an attacker requests a high number of keys. The victim spends considerable amount of computing resources doing useless modular exponentiation

B. Station-To-Station (STS) Protocol

Diffie-Hellman key exchange is vulnerable to a man-in-the middle attack. To prevent this problem, the Station-To-Station (STS) protocol was proposed in [6]. To add authentication, the STS protocol requires both the parties to have a pair of public keys for signature generation and verification [7], and to know a publicly released symmetric key encryption, it encrypts the signatures with the session key subsequently to show the knowledge of this session key. However, signatures and certificates cause the messages to increase considerably in size.

Then, the protocol goes as follows:

(1) Alice generates a random number, x, and sends it to Bob.
(2) Bob generates a random number, y. Using the Diffie-Hellman protocol he computes their shared key based on x and y: k. He signs x and y, and encrypts the signature using k. He then sends that, along with y, to Alice. y, $Ek(SB(x, y))$.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

(3) Alice also computes *k*. She decrypts the rest of Bob's message and verifies his signature. Then she sends Bob a signed message consisting of *x* and *y*, encrypted in their shared key. *E*k (SA(*x*, *y*)).
(4) Bob decrypts the message and verifies Alice's signature.
It is vulnerable to unknown key share attacks [8].

C. Encrypted Key Exchange Protocol

The Encrypted Key Exchange (EKE) protocol was designed by Steve Bellovin and Michael Merritt [9]. It provides security and authentication on computer networks, using both symmetric and public-key cryptography in a novel way: – A shared secret key is used to encrypt a randomly generated public key.

Then, the protocol goes as follows:

(1) Alice generates a random public-key/private-key key pair. She encrypts the public key, *K´*, using a symmetric algorithm and *P* as the key: *EP*(*K´*). She sends Bob *A*, *E*P(*K´*)
(2) Bob knows *P*. He decrypts the message to obtain *K´*. Then, he generates a random session key, *K*, and encrypts it with the public key he received from Alice and *P* as the key. He sends Alice *EP*(*EK´*(*K*))
(3) Alice decrypts the message to obtain *K*. She generates a random string, *R*A, encrypts it with *K*, and sends Bob – *E*K(*R*A)
(4) Bob decrypts the message to obtain *R*A. He generates another random string, *R*B, encrypts both strings with *K*, and sends Alice the result. *E*K(*R*A, *R*B)
(5) Alice decrypts the message to obtain *R*A and *R*B. Assuming the *R*A she received from Bob is the same as the one she sent to Bob in step (3), she encrypts *R*B with *K* and sends it to Bob. *E*K(*R*B)
(6) Bob decrypts the message to obtain *R*B. Assuming the *R*B he received from Alice is the same one he sent to Alice in step (4), the protocol is complete.

Both parties now communicate using *K* as the session key.

EKE can be implemented with a variety of public key algorithms: RSA [10], ElGamal [11], Diffie-Hellman [3].

D. Fortified Key Negotiation

Key exchange schemes such as Diffe-Hellman are vulnerable to man-in-the-middle attacks, and thus are often augmented by means of shared secrets. Where these secrets must be memorized, they will usually be vulnerable to guessing attacks [12]. We show how collision rich hash functions [13] can be used to detect such attacks while they are in progress and thus frustrate them. It uses a hash function of two variables that has a very special property:

It has many collisions on the first variable while having effectively no collisions on the second variable.

• $H´(x, y) = H(H(k, x) \bmod 2m, x)$,

where $H(k, x)$ is an ordinary hash function on *k* and *x*.

This scheme protects key-negotiation schemes from poorly chosen passwords [14, 15] and man-in-the-middle attacks. Here's the protocol. Alice and Bob share a secret password, *P*, and have just exchanged a secret key, *K*, using Diffie-Hellman key exchange. They use *P* to check that their two session keys are the same (and that Eve is not attempting a man-in-the-middle attack), without giving *P* away to Eve.

(1) Alice sends Bob
H´ (*P*, *K*)
(2) Bob computes *H´* (*P*, *K*) and compares his result with what he received from Alice. If they match he sends Alice *H´*(*H*(*P*, *K*))
(3) Alice computes *H´* (*H*(*P*, K)) and compares her result with what she received from Bob.

If Eve is trying a man-in-the-middle attack, she shares one key, *K*1, with Alice, and another key, *K*2, with Bob.

- To fool Bob in step (2), she has to figure out the shared password and then send Bob *H´* (*P*, *K*2).
- With a normal hash function she can try common passwords until she guesses the correct one, and then successfully infiltrate the protocol.
- But with this hash function, many passwords are likely to produce the same value when hashed with *K*1.
- So when she finds a match, she will probably have the wrong password, and hence Bob in step 2 will not be fooled when he computes his own *H´* (*P*, *K*).

E. Shamir's Three-Pass Protocol

This protocol enables 2 parties to communicate securely (over 3 message exchanges) with each other without the need for any advance exchange of either secret keys or public keys [16]. There is no agreement or exchange of keys. This protocol employs a commutative cipher, where:

$$E_A[E_B(P)] = E_B[E_A(P)]$$

This protocol makes use of RSA key algorithm does work and is feasible. It achieves mutual explicit authentication and mutual entity authentication A encrypts message with her secret key, A.

$(M)_A = C1$

B encrypts the message received from A, with his secret key B.

$[(M)_A]_B = C2$

A decrypts the message received (C2) with her key and sends:

$\{[(M)_A]_B\}_{A'} = \{[(M)_B]_A\}_{A'} = (M)_B = C3$

B decrypts the message with his key to recover the message.

This protocol is subject to a man-in-the-middle attack. Like the Diffie Hellman protocol, its security is dependent on the intractability of the discrete logarithm problem.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

F. Interlock Protocol

Most cryptographic protocols rely on the prior establishment of secret or public keys or passwords. However, the Diffie-Hellman key exchange protocol introduced the concept of two parties establishing a secure channel (that is, with at least some desirable security properties) without any such prior agreement. Unauthenticated Diffie-Hellman, as an anonymous key agreement protocol, has long been known to be subject to man in the middle attack. However, the dream of a "zipless" mutually authenticated secure channel remained.

The Interlock Protocol [17] was described as a method to expose a middle-man who might try to compromise two parties that use anonymous key agreement to secure their conversation. The Interlock protocol works roughly as follows: Alice encrypts her message with Bob's key, then sends half her encrypted message to Bob. Bob encrypts his message with Alice's key and sends half of his encrypted message to Alice. Alice then sends the other half of her message to Bob, who sends the other half of his. The strength of the protocol lies in the fact that half of an encrypted message cannot be decrypted. Thus, if Mallory begins her attack and intercepts Bob and Alice's keys, Mallory will be unable to decrypt Alice's half-message (encrypted using her key) and re-encrypt it using Bob's key. She must wait until both halves of the message have been received to read it, and can only succeed in duping one of the parties if she composes a completely new message.

The problem with this protocol is it only usable when both parties must be communicating in real time and it is subject to replay attack[18].

V. ADVANTAGES AND DISADVANTAGES OF ASYMMETRIC KEY DISTRIBUTION

Public key cryptography offers additional features that are not easily obtainable with symmetric cryptography which are Non repudiation, True data origin authentication. It does not require Online Trusted Server. No need for secrecy of encryption keys (public keys). Only the private key must be kept secret (authenticity of public keys must, however, be guaranteed).In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario. Depending on the mode of usage, a public/private key pair may be not need to be changed for a considerable amount of time. Problems with this type of key distribution are they are slow compared to symmetric cryptography. Mathematically computations used to encrypt data require more time. Key sizes are typically much larger than those required for symmetric-key encryption, and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques. In most cases, a pre-existing public key infrastructure is required where a CA, RA, CRLs etc are needed. Unlike symmetric keys (which are usually random numbers of a certain length), asymmetric keys are numbers with special mathematical properties that require a key generation algorithm. No public-key scheme has been proven to be secure (the same can be said for block ciphers). The most effective public-key encryption schemes found to date have their security based on the presumed difficulty of a small set of number-theoretic problems. An encrypted message can only be sent to a single recipient. It is not feasible to send to more than 1 entity as a recipient's private key is used to decrypt a message.

VI. CONCLUSION

We explained the need of session key and how to exchange it with symmetric and asymmetric cryptography. We have presented a study on different asymmetric authentication protocols. Furthermore, we have discussed the benefits and problems with asymmetric key distribution algorithms. All algorithms are having the benefits and as well as the problems. So based on our application choose the suitable authentication protocol.

REFERENCES

[1] S. Blake-Wilson, D. Johnson, A. Menzes, "Key exchange protocols and their security analysis," in Proc of 6th IMA International Conference on Cryptography and Coding, pp-30-45, Dec 17-19, 1997.

[2] Pushpendra Kumar pateriya Srijith S. S. Kumar, "Analysis on Man in the Middle Attack on SSL ", IJCA journal, vol. 45, no 23, 2012.

[3] W.Diffie, M. Hellman, "New directions in cryptography", IEEE Transaction on Information Theory, vol. 22 no. 6, pp, 644-654, Nov, 1976.

[4] Vishal Garg et al,"Improved Diffie Hellmen algorithm for Network security Enhancement ", International journal of computer Technology and Applications, vol 3 (4), 2012.

[5] Sattar J Aboud, Abid T. Al Ajeeli Cryptanalysis of an advanced authentication scheme, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 10, 2011.

[6] Simon Blake-Wilson and Alfred Menezes, Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol.

[7] ShaikRiyaz, J.Rajkala, M.Ramakrishna "Data security and Authentication using Stegonography and STS protocol" International Journal of Advanced Research in computer science and Electronic Engg., vol 1, Issue 5, 2012.

[8] BURTON S. KALISKI, "An Unknown Key-Share Attack on the MQV Key Agreement Protocol", JR.RSA Laboratories, ACM Transactions on Information and System Security, Vol. 4, No. 3, pp. 275–288, August 2001.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

[9] Steve Bellovin and Michael Merritt, Augmented encrypted key exchange, First Conf. Computer & Commn. Security, USA.

[10] Mollin," RSA and Public-Key Cryptography" Journal of the ACM, Vol. 51, No. 2, March 2004.

[11] Allam Mousa, "Security and Performance of ElGamal Encryption Parameter, "Journal of Applied Sciences 5, Asian network for Scientific Information, 883-886, 2005.

[12] Mathieu Baudet, Deciding Security of Protocols against Offline Guessing Attacks, proceedings of the ACM CCS'05 conference, Alexandria, Virginia, USA, Nov. 7–11, 2005.

[13] Necessary & sufficient conditions for collision free hashing Alexander Russell, NSF grant 92-12184, AFOSR F49620-92-J-0125 and DARPA N00014-92-J-1799.

[14] Anderson, fortifying key negotiation schemes with poorly chosen passwords, Vol 30, Issue 13, Jun 1994.

[15] R J Anderson and T M a Lomas, On fortifying key negotiation schemes with poorly chosen passwords, University Computer Laboratory Pembroke Street, Cambridge CB2 3QG.

[16] Yoshito kanamori and seong-moo yoo, Quantum three-pass protocol: key distribution using quantum superposition states, International Journal of Network Security & its Applications (IJNSA), Vol. 1, No. 2, Jul 2009.

[17] Steve Bellovin and Michael Merritt, An attack on the Interlock Protocol, when used for authentication, IEEE transactions on Information Theory, Vol. 40, No. 1, Jan 1994.

[18] Villalba, J Commun Preventing replay attacks on speaker verification systems, IEEE International Carnahan Conference, 2011.

[19] ANSI X9.30 (Part 1), Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry {Part 1: The Digital Signature Algorithm (DSA)}, 1995.

[20] ANSI X9.31, Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), Working Draft, Mar 1998.

[21] ANSI X9.62, The Elliptic Curve Digital Signature Algorithm (ECDSA), Working Draft, Aug 1998.

[22] ANSI X9.63, Elliptic Curve Key Agreement and Key Transport Protocols, Working Draft, Oct 1998.

[23] Kuljeet Kaur, Fortification of Transport Layer Security Protocol, Special Issue on Network Security and Cryptography, IJCA, NSC–2011.

Author's Profile

Dr. D. S. R. Murthy obtained B. E. (Electronics) from Bangalore University in 1982, M. Tech. (CSE) from Osmania University in 1985 and Ph.D. (CSE) from JNTUH, Hyderabad in 2011. He is presently working as Professor in Information Technology, SNIST, and Hyderabad since Oct. 2004. He earlier worked as Lecturer in CSE, NIT (formerly REC), Warangal, India during Sep. 1985 – Feb. 1993, as Assistant Professor in CSE, JNTUCE, Anantapur, India during Feb. 1993 – May 1998, as Academic Coordinator, ISM, Icfaian Foundation, Hyderabad, India during May 1998 – May 2001 and as Associate Professor in CSE, SNIST during May 2001 - Sept. 2004. He worked as Head of the Dept. of CSE, JNTUCE, Anantapur during Jan. 1996 – Jan 1998, Dept. of IT, SNIST during Apr. 2005 – May 2006, and Oct. 2007 – Feb. 2009. He is a Fellow of IE (I), Fellow of IETE, Senior Life Member of Computer Society of India, Life Member of ISTE, Life Member of SSI, DOEACC Expert member, and Chartered Engineer (IE (I) & IETE). He is Chief Superintendent of Examinations (Autonomous), SNIST, and Hyderabad since February 2011. He is Board of Studies Member of SRKPG College (Autonomous), Nandyal, SKD University since August 2010 and Anurag Group of Institutions (Autonomous), Hyderabad, JNTUH since July 2012. He is a Reviewer of International Journal of Advanced Research in Computer Science (IJARCS), International Journal of Computational Intelligence and Information Security (IJCIIS), International Journal of Computer Science and Information Technology (IJCSIT), International Journal of Advanced Computer Science and Applications (IJACSA), Science Journal of Electrical & Electronic Engineering (SJEEE), International Journal of Network Security (IJNS), International Journal of Computer and Information Technology (IJCIT), Honorary Peer Reviewer of Global Journal of Researches in Engineering (GJRE), Advisory Board Member of International Journal of Engineering and Innovative Technology (IJEIT), Editorial Board Member of World Academy of Research in Science and Engineering (WARSE), International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Journal of Computer Science and Engineering (JCSE), International Journal of Computer Applications and Information Technology (IJCAIT). He is a Member of International Association of Computer Science and Information Technology (IACSIT). He published a text book on C Programming & Data Structures. His research interests are Image Processing, Image Cryptography and Network Security. He published research papers in International Journal of Computer and Network Security (IJCNS), International Journal of Computer Theory and Engineering (IJCTE), International Journal of Computational Intelligence and Information Security (IJCIIS) and in International Journal of Advanced Research in Computer Science (IJARCS).

Mrs. B. Madhuravani, Associate Professor, Department of CSE, Malla Reddy College of Engineering & Technology, Maisammaguda, Dulapally, Secunderabad, affiliated to JNTUH, Hyderabad, Andhra Pradesh, India. She received M. Tech in Computer Science & Engineering from JNTU, Hyderabad in 2010. Her total teaching experience is 7 years. Her research interests include Computer Networks, Network Security, Distributed Systems and Data Structures.

Mrs. G. Sumalatha, Asst. Professor, Department of Information Technology SreeNidhi Institute of Science and Technology (An Autonomous Institution) Hyderabad – 501 301, Andhra Pradesh, India. She received M. Tech in Computer Science & Engineering from JNTU, Hyderabad in 2010. Her total teaching experience is 7 years. Her research interests include Computer Networks, Network Security, and Theory of computation.