International Journal of Scientific Research in Computer Science and Engineering

Research Paper

Volume-1, Issue-2, March-April-2013

Available online at www.isroset.org

ISSN No. 2320-7639

Denial of Services Based Countermeasures Key Exchange Protocols

Arpit Agrawal

Department of Computer Engineering
Institute of Engineering & Technology
Devi Ahilya University
M.P., India
arpit_agrw@yahoo.co.in

Jitendra Soni*

Department of Computer Engineering
Institute of Engineering & Technology
Devi Ahilya University
M.P., India
jitendrasoni2008@yahoo.co.in

Received: 26 Feb 2013 Revised: 12 Mar 2013 Accepted: 10 Apr 2013 Published: 30 Apr 2013

Abstract-Cryptography has become indispensable in areas such as e-commerce, the legal safeguarding of medical records, and secures electronic communication. Cryptographic protocols like key exchange usually require expensive computations such as finite field exponentiations or elliptic curve scalar-point multiplications. In an adversarial environment such as the Internet, an attacker could effect a denial of service attack against a server by forcing the server to perform many instances of a cryptographic protocol. To help tackle this challenge, several cryptographic primitives and constructions have been proposed, including mechanisms to securely distribute data over a unidirectional insecure channel, schemes in which leakage of cryptographic keys can be traced back to the leaker, and techniques to combine revocation and tracing capabilities. In this paper we proposed the integration of denial of service countermeasures key exchange protocol by introducing a formal model for denial of service resilience that complements model for secure key agreement, we cover a wide range of existing denial of service attacks and prevent them by carefully using client queries and puzzles.

Keywords- Cryptography, DoS, Secure Key Exchange Protocol and Digital Signature.

Introduction

Cryptography is the study of methods for sending messages in a secret, namely in enciphered form, so that only the intended recipient can remove the disguise and read the message. Cryptography is not only about encrypting and decrypting messages; it is also about solving real-world problems that require information security. There are four main requirements that need to be satisfied:

Confidentiality: Eve should not be able to read Alice's message to Bob. The main tools are encryption and decryption algorithms.

Data Integrity: Bob wants to be sure, that Alice's message has not been altered. For example, transmission errors might occur. Also an adversary might intercept the transmission and alter it before it reaches the intended recipient. Many cryptographic primitives, such as hash functions, provide methods to ensure data integrity despite malicious or accidental adversaries.

Authentication: Bob wants to be sure that only Alice could be able to send the message he received. Under this heading we also include identification schemes and password protocols. There are actually two types of authentication that arise in cryptography: entity authentication and data-origin authentication. Often the term identification is used to specify entity authentication, which is concerned with providing the identity of the parties involved in a communication. Data-origin authentication focuses on tying the information about the origin of the data, such as the creator and time of creation, with the data [1-2].

Non-repudiation: Alice cannot claim she did not send the message. Non-repudiation is particularly important in electronic commerce applications, where it is important that a consumer cannot deny the authorization of a purchase. The basic problem of password-authenticated key exchange is as follows. Two parties wish to authenticate each other and agree upon a secret key, but they only share a short text string, not a long cryptographic secret. The client cannot simply tell the

server her password as proof of identity because a malicious "phishing" server could then steal the password and impersonate the client at the real server. A password-authenticated key exchange protocol allows each party to authenticate the other's identity based solely on their knowledge of a short password, without revealing any useful information about the password to any other party; moreover, the two parties can also agree on a shared key suitable for other cryptographic purposes such as bulk encryption [5] and [7].
In this paper we proposed a denial of service countermeasures key exchange protocol for denial of service resilience that complements model for secure key agreement attacks detection and prevention system.

Background Techniques

Symmetric Key Algorithms
In symmetric key algorithms the encryption and decryption keys are known to both, Alice and Bob. For example, the encryption key is shared and the decryption key can be easily calculated from it. In many cases, the encryption key and the decryption key are the same. The following figure shows the principle of symmetric ciphers.

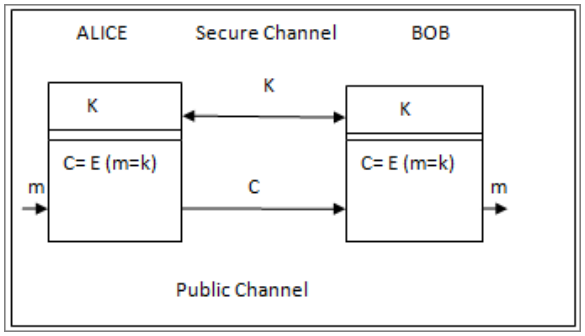

Figure 1: Principle of Symmetric Ciphers

Alice encrypts the secret message using the symmetric key k. The cipher text is transmitted via a public channel to Bob, who is able to transform the cipher back to the plaintext with the help of the symmetric key k, which has to be interchanged in a secure way before the communication starts. The security of the system is based on the confidentiality of the used key. For all $m \in M$, $k \in K$ it has to hold that $D(E(m, k), k) = m$

Cryptographic Hash Functions
A cryptographic hash function H takes an input of arbitrary length and produces a message digest, i.e. a fingerprint of the message of a fixed length (e.g. 128 or 160 bits). Such a function has to satisfy certain properties:

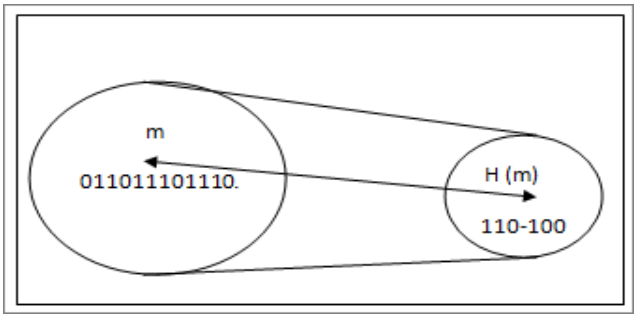

Figure 2: Cryptographic Hash Functions

For a given message m, the message digest H(m) can be calculated very quickly. Weak Collision Resistance: Given a fingerprint y, it is computationally infeasible to find an m with H(m) = y. Strong Collision Resistance: It is computationally infeasible to find messages m1 and m2 with H(m1) = H(m2).

Note that, since the set of possible messages is much larger than the set of possible hash functions, there are always many examples of messages m1 and m2 with H(m1) = H(m2). The last requirement claims that it should be hard to find such examples.
Examples
→ MD2, MD4, MD5 (128 bit)
→ SHA-1 (160 bit)
→ RIPMED-160 (160 bit)
→ SHA-256, SHA-384, SHA-512 (256,384 and 512 bit)

If a hash function is conditioned on a secret key k, we call it Message Authentication Code (MAC), or keyed hash function. If Bob receives a MAC, he is able to verify its authenticity by using the symmetric key k.

Public Key Algorithms
Public Key algorithms have been introduced in the 1970s and revolutionized cryptography. Suppose Alice wants to securely communicate with Bob, but they are hundreds of kilometers apart and have not agreed on a key to use. One solution would be to assign a trusted courier to carry the key from one to the other, but this is simply not always possible. Public key cryptography offers a solution, where the encryption key is made public but it is computationally infeasible to find the decryption key without information known only to Bob [3-4].

One-Way Functions
A One-Way Function is a function which is easy to compute, but hard to invert. The terms easy and hard can mean that the computation of the function in the forward direction takes some seconds, but to compute the inverse could take some months, if possible at all. A trapdoor one-way function is a one-way function for which the inverse direction is easy, if a certain piece of information is given (the trapdoor), but difficult otherwise. Practice-oriented systems are based on the following one-way functions.
→Multiplication and factoring
→ Modular squaring and square roots
→ Discrete exponential and logarithm
→Subset Sum Problem

Hybrid Crypto-Systems
Symmetric and asymmetric key algorithms offer a whole set of advantages; however they do have certain disadvantages. Symmetric ciphers have a very simple design and offer high transfer rates. On the other hand, a secret key has to be exchanged in advance. Asymmetric key algorithms dispose the key exchange problem, but the data rates are much lower. Therefore, the idea of combining both algorithms is almost self-evident. Hybrid Crypto Systems realize an asymmetric key exchange to create a symmetric session key. The basic build up is depicted.
At the beginning of the communication a symmetric session key k is negotiated. A typical strategy is to generate random numbers, which are exchanged via asymmetric communication and which are concatenated, so that every communication partner contributes parts of the key. When this process is finished, a simple symmetric cipher like the AES can be used to exchange the secret information [2-4].

Digital Signatures
When Alice sends a message to Bob using a symmetric cipher, Bob can be sure that it stems from Alice, since she is the only one who is in possession of the secret key. However, Bob cannot prove to a third party that he received the message from Alice - she could simply claim it was Bob on his own who compiled it.
When using asymmetric ciphers, everybody who fetches Bob's public key could send him a message. This way, confidentiality is ensured, while authenticity is not at hand. What is missing is a liaison between the sender and the message, which can be realized using Digital Signatures. As equivalent to a personal signature, digital signatures have to satisfy some conditions.
Digital Signatures:
→have to be dependent on message and sender,
→ can only be generated by the signer and
→ can be verified by every user of the system.

Certificates

With the help of digital signatures the integrity of the message is ensured. A problem that still occurs is the lack of evidence that the public key PKx really belongs to user X. To counter this aw certificates are implemented. Certificates bind a public key to a particular person. A trusted third party, we call it Certification Authority CA, signs the data with its own secret key SKCA. Now only the public key of the CA PKCA has to be authentically established within the system.

Basic Key Exchange Models:

The basic idea of the model is as follows. We envision a number of honest, interacting parties connected by communication links. The communication links are controlled by an adversary. The adversary can observe messages sent by the parties, which models passive eavesdropping. The adversary can also modify, delete, or reorder messages sent by the parties, and can inject her own messages as well, which models active interference in the network. In addition to controlling the communication links, the adversary is also allowed to learn certain information from parties, which models partial compromise of private information.

To analyze security in this model, we define one or more tasks, games, which the adversary needs to accomplish in order to have broken the security of the protocol, provided that the adversary has not compromised particular interacting parties so much so to make the tasks trivial. We now describe the BPR model more precisely [5-7].

Model setup

Participants, Each interacting party is either a client or a server, is identified by a unique fixed length string, and the identifier is a member of either the set Clients or Servers, respectively, with Parties = Clients Servers- Passwords. Authentication secrets are short strings selected uniformly at random from an appropriate set. Passwords are chosen from the set Passwords. Typically, this set is large, but not so large that a brute-force search is infeasible.

Session key security

Informally, a session key output by a key exchange protocol is considered secure if it is indistinguishable from a random string chosen from the same distribution as the possible session keys. In order to model session key security, the BPR model defines an additional query that is used to define the task of the adversary, which is to distinguish a session key from a random string for an uncompromised session. The Test query, which does not correspond to any real-world operation of the adversary, is used to define the game that the adversary must win in order for the protocol to be considered broken [6-8].

Challenge Response Authentication

Challenge-Response protocols serve as secure authentication processes for instances which are based on knowledge. A trusted authority sends a random challenge to a user who wants to authenticate himself. He generates a response and sends it back to the authority which verifies if the authentication was successful or not. The big advantage over existing password protocols is that the challenge is not fixed but varies between different authentication attempts. With this, previously recorded responses will not help a potential attacker [1-3] and [8].

Proposed Technique

Now, we described Denial of service countermeasures key exchange protocol. Our Denial of service countermeasures key exchange protocol, given in Figure 1, is a secure authenticated key exchange protocol. We use the problem of finding pre-images for a random hash function as the expensive puzzle at the heart of the puzzling relation that a client needs to solve. We designed Denial of service countermeasures key exchange protocol by modifying the pre-session to include a denial of service countermeasure based on hash functions.

Denial of service countermeasure: A client wishing to prove its legitimate intention must solve a computational puzzle that involves finding a pre-image in a hash function. Differing from previous approaches, we include the client identity, server identity, and client ephemeral public key in the hash function input. This ties the puzzle solution to a particular session so it cannot be used for other purposes.

Session key security: For key agreement, we use the protocol without significant modification. It combines the static and ephemeral keys of each party at various stages through the protocol to achieve strong security and authentication. By hashing these values together in the early stages of the protocol, the protocol attains resilience against ephemeral private key leakage without using non-standard assumptions such as the Knowledge of Exponent Assumption used in the

argument. By combining these in the computation of the session key, implicit authentication is achieved since only a party who knows both the static and ephemeral private keys can construct the correct session key. The security argument rests on the Gap Diffie-Hellman assumption and the random oracle model.

Protocol specification

The Denial of service countermeasures key exchange protocol is given in Figure 3 below. Notation Denial of service countermeasures key exchange protocol operates over a finite cyclic group G for which the Gap Diffie-Hellman (GDH) assumption holds. Let λ be a security parameter. The notation L[i] denotes the ith component in the tuple L: for example, if ch = (i, j), then ch[1] = i and ch[2] = j. H0 and H1 are random hash functions that return bit strings; all other hash functions are random hash functions that return integers between 1 and q -1, where q is the order of the group generated by g. We use the notation x[1&..w] to denote the first w bits of x. Note that according to our definition the protocol starts by establishing the session. The steps before the establishment of a session are the pre-session. For our protocol as described in Figure 1, the effects of the information compromise queries of the adversary are as follows:

Reveal-Static-Key (A): returns client A^'s long-term private key a; this is's certified private information.
Reveal-Static-Key (B): returns server B's long-term private key b, this is B's certified private information.
Reveal-Ephemeral-Key ([A, B, ch, *]): returns client A's ephemeral private key x.
Reveal-Ephemeral-Key ([B, A, ch, re]): returns server B's ephemeral private key y.
Reveal-Session-Key (sid): returns the session key K.
DoS-Expose (B): reveals server B's DoS private value ρ; this is ^B's non-certified private information.
The session identifier held by the client A is [A, B, ch, re], and the session identifier held by the server B is [B, A, ch, re].

In our Denial of service countermeasures key exchange protocol can be tuned by the server based on the load it experiences. The client must find a pre-image for the hash function H1; for concreteness, we have specified that this should have 20 bits of output, but the length could be a parameter w set by the server depending on its current load. The server would need to include w in the computation of j on line 2, return w as part of ch on line 3, and include w in the check on line 9.

Client A		**Server B**
G, a, A= g^a,B		g, b, B, = g^b, A, ρ∈R {0, 1}
	"Hello", A, B	I ∈R $\{0, 1\}^\lambda$
		J=H_0(ρ, A, B, i)
x=∈R$\{0, 1\}^\lambda$	ch	ch =(i, j)
x = H2(x; a)		
X = g^x		
find l s.t. H1(A, B, ch, X, l)[…….20] = 0…….. 0		
re = (X, l`)		
establish session [A, B, ch, re]		
Ψ= (ch; re)	A, ch, re	verify ch[2] = H0(ρ, A, B, ch[1])
	verify H1(A, B, ch, re)[1…….20] = 0……. 0	
		establish unique session [B, A, ch, re]
		store A, Ψ= (ch, re)
		X = re[1]

			verify $X \in G$
			$y \in R \{0, 1\}^{\lambda}$
			y = H2(y, b)
			store $Y = g^{y}$
			d = H3(X, A, B)
			e = H3(Y, A, B)
			$\sigma = (XA^{d})^{y+eb}$
		store M1 = H4("server finished"; A, B, ch, re, Y, σ)	
		store M2 = H4("server finished"; A, B, ch, re, Y, σ)	
	verify $Y \in G$	Ψ, Y, M	store K = H5(A, B, ch, re, Y, σ)
d = H3(X, A, B)			
e = H3(Y, A, B)			
$\sigma = (Y B^{e})^{x+da}$			
Verify M1			
M2 = H5("client finished", A, B, ch, re, Y, σ)			
K = H5(A, B, ch, re, Y, σ)		Ψ, M2	Verify M2

Figure 3: proposed Denial of service countermeasures key exchange protocol Scheme

As for the efficiency of the denial of service countermeasure we have added, it follows from our denial of service resilience analysis that the server is not significantly burdened by this addition. The burden on the client is substantial, but this is by design: the reasoning behind the denial of service countermeasure is that the client must commit significant computational resources if the connection request is to be perceived as legitimate.

Security analysis of Denial of service countermeasures key exchange protocol

Now we analysis the Denial of service countermeasures key exchange protocol is a secure authenticated key exchange protocol. The argument follows from the argument presented for CMQV. We relate the difficulty of distinguishing the session key of a fresh session to the difficulty of solving the Computational Diffie-Hellman problem in a group where the Gap Diffie-Hellman assumption holds.

If H0…….,H5 are random oracles, and G is a group where the Gap Diffie-Hellman assumption holds, then the Denial of service countermeasures key exchange protocol protocol is a secure key exchange protocol.

We argue that the security of Denial of service countermeasures key exchange protocol follows as: Verifying condition 1 is straightforward. It remains to verify condition 2. We note that in our model, parties possess additional (uncertified) private information ρ, which the adversary can obtain via DoS Expose query.

We construct a polynomial bounded algorithm S that transforms an attack on Denial of service countermeasures key exchange protocol into a solution to a Gap Diffie-Hellman instance. It does this by establishing and simulating parties as in the analysis. The only difference is that when parties are established the solver selects randomly the value ρ for each party. The DoS Expose queries are answered faithfully and they do not affect the freshness of the session. Since this new query is not relevant to the security analysis, S can transform the Denial of service countermeasures key exchange protocol.

Denial of Service Resilience Analysis

We show that Denial of service countermeasures key exchange protocol is denial-of-service-resilient. Since this definition (and the subsequent definition of a puzzling relation) includes the intentionally vague terms "expensive operation", "easy", and "hard", we need to define what these terms mean for a concrete instantiation of the Definition. For our purposes, an expensive operation is one of the following operations: storing a per-connection or per-session value in memory (other than a long-term value), performing a group exponentiation, or making a large number of calls (say, more than 2^{10}) to a hash oracle.

We first argue that the hash function construction we use is a puzzling relation, then that Denial of service countermeasures key exchange protocol is a denial-of-service-resilient protocol using that puzzling relation.

Deciding membership in R is easy for a particular tuple because it involves only a single call to H1. Given A, B, ch, we need to produce re such that H1(A, B, ch, re)[1……..20] = 0…… 0.

To find such an re requires finding a preimage for the random hash function. The oracle U helps us find other preimages of H1, but we cannot ask U for preimages involving (A, B, ch). Our task, then is to find a preimage of the correct format involving A, B , and ch. Since H1 is a random hash function outputting 20 bits, this is a hard task that requires approximately 2^{20} queries on average. This hash puzzle is similar to the partial inversion proof of work (PIPOW) problem and the exact tradeoff between work and probability of success can be calculated. In practice, H1 as used in the puzzling relation could be implemented by using a standard cryptographic hash function, such as SHA-1, and truncating the output to the first w bits. The puzzle used in our Denial of service countermeasures key exchange protocol could be tuned by the server based on the load it experiences. In times of light load, the server could require that clients truncate only to the first 5 or 10 bits of output, but in heavier load could require that clients truncate to 20 or 25 bits of output to make the cost of mounting a denial of service attack higher. For w = 20, it takes just under 3 seconds to perform 2^{20} SHA-1 evaluations on one core of our 2 GHz Intel Core 2 Duo processor. This may be an acceptable computational burden for the client in many scenarios.

Conclusion

Key exchange is an important cryptographic primitive typically used for building secret channels between two parties. A designing and analyzing key exchange protocol is a non-trivial task. Denials of service (DoS) attacks, in which many client nodes computers attack a single server node, are of significant concern on the Internet today. Distributed attacks are very difficult to defend against. In this paper, we described other constructions for achieving denial of service resilience in key exchange protocols. We also discussed and analysis implementing the puzzling relation with other types of puzzles. Our proposed secure key exchange protocol is more efficient than existing protocols.

References

[1] N H Ayachit, Santosh L Deshpande, and Kamakshi Prasad V, "Evolutionary Computing based secure key management Protocol", IEEE 2010.

[2] Abdelouahid Derhab and Nadjib Badache. A Self-Stabilizing Leader Election Algorithm in Highly Dynamic Ad Hoc Mobile Networks. IEEE Transactions on Parallel Distributed Systems, 19(7), pp- 926–939, 2008.

[3] Lin Yao, Bing Liu, Kai Yao, Guowei Wu, and Jia Wang, "An ECG-Based Signal Key Establishment Protocol in Body Area Network", IEEE 2010 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, pp. 233-238.

[4] H. Wen, P.-H. Ho, C. Qi, and G. Gong, "Physical layer assisted authentication for distributed ad hoc wireless sensor networks", IEEE 2010, Special Issue on Multi-Agent & Distributed Information Security, pp. 390-396.

[5] CAO Zheng, YIN Pengpeng, and LU Zhengjun, "A Key Management Scheme for Multicast Based on Layer 2 Control", IEEE 2010, International Conference on Computer and Information Technology (CIT 2010), pp. 1512-1518.

[6] Jinxia Yu, Yongli Tang, Xiancha Chen and Wenjing Liu, "Choice Mechanism of Proposal Distribution in Particle Filter", IEEE 2010, Proceedings of the 8th World Congress on Intelligent Control and Automation, pp. 1051-1056.

[7] Jinxia Yu, Yongli Tang, Xiancha Chen and Wenjing Liu, "Choice Mechanism of Proposal Distribution in Particle Filter", IEEE 2010, Proceedings of the 8th World Congress on Intelligent Control and Automation, pp. 1051-1056.

[8] FengHe and Kuan Hao, Hao Ma, "S-MAODV:A Trust Key Computing Based Secure Multicast Ad-hoc On Demand Vector Routing Protocol", IEEE 2010, pp. 434-438.

