PROTECTING KEY EXCHANGE AND MANAGEMENT PROTOCOLS AGAINST RESOURCE CLOGGING ATTACKS

Rolf Oppliger
Swiss Federal Office of Information Technology and Systems (BFI)
IT Security Group
Monbijoustrasse 74, CH-3003 Berne, Switzerland
rolf.oppliger@mbox.bfi.admin.ch

Abstract Many cryptographic key exchange and management protocols involve computationally expensive operations, such as modular exponentiations, and are therefore vulnerable to resource clogging attacks. This paper overviews and discusses the basic principles and the rationale behind an anti-clogging mechanism that was originally designed and proposed to protect the Photuris Session Key Management Protocol against resource clogging attacks. The mechanism was later approved by the IETF IPsec WG to be included into the Internet Key Management Protocol (IKMP) or Internet Key Exchange (IKE) protocol respectively. The paper introduces and discusses the Photuris anti-clogging mechanism, derives some design considerations, and elaborates on possibilities to use similar techniques to improve an existing HTTP state management protocol and to protect TCP/IP implementations against TCP SYN flooding attacks.

Keywords: Photuris Session Key Management Protocol, Internet Key Management Protocol (IKMP), Internet Key Exchange (IKE), resource clogging attacks, anti-clogging mechanism, anti-clogging token, HTTP state management, TCP SYN flooding attacks.

1. INTRODUCTION

To meet the security requirements of a steadily increasing number of Internet users, the Internet Engineering Task Force (IETF) chartered an IP Security (IPsec) Working Group (WG) to develop and standardize an IP Security Protocol (IPSP) and a corresponding Internet Key Management Protocol (IKMP). In August 1995, the IETF IPsec WG published a series of Request for Comment (RFC) documents (RFCs 1825 to 1829) that specified a preliminary version of IPSP [Atk97,Opp98a] and the Internet Engineering Steering Group (IESG) approved this protocol specification to enter the Internet standards track as a Proposed Standard.

Meanwhile, the participants of the IETF IPsec WG have been working hard to further refine the IP security architecture, and to finish up the IPSP and IKMP specifications accordingly. More recently, the IP security architecture and the IPSP and IKMP were specified in RFCs 2401 to 2412 as well as RFC 2451, and the IKMP was renamed to be further referred to as the Internet Key Exchange (IKE) protocol. As such, it combines the Internet Security Association and Key Management Protocol (ISAKMP) and the OAKLEY Key Determination Protocol within the IPsec Domain of Interpretation (DoI). In short, the ISAKMP provides a framework for authentication and key exchange protocols, whereas the OAKLEY Key Determination Protocol actually performs an authenticated key exchange in this framework (which is basically a Diffie-Hellman key exchange with a subsequent authentication step).

The IETF IPsec WG started from the insight that any key exchange or management protocol that is to scale for the global Internet must make use of public key cryptography. However, the IETF IPsec WG soon realized that there is a potential Achilles heel in any such protocol. The Achilles heel is due to the fact that the use of public key cryptography requires computationally expensive operations, such as modular exponentiations, and that a corresponding key exchange or management protocol is therefore vulnerable to resource clogging attacks. Note that if an attacker can initiate several simultaneous key exchange or management protocol executions that each take some time and memory to perform, the victim's resources will be clogged very rapidly. To make things worse, the attacker can initiate the protocol executions with randomly chosen IP source addresses (to stay anonymous). Consequently, resource clogging attacks represent a significant class of denial-of-service or degradation-of-service attacks that must be considered with care.

Taking the feasibility of resource clogging attacks into account, some form of protection is required for any key exchange or management protocol that is going to be used on a large scale. This is equally true for the IKE protocol and for any other key exchange and management protocol (not necessarily based on IP).

The aim of this paper is to elaborate on possibilities to protect key exchange and management protocols against resource clogging attacks. Consequently, the paper overviews and discusses an anti-clogging mechanism that has been developed within the IETF IPsec WG to protect the IPsec suite of security protocols in Section 2. The mechanism includes an initial message exchange that is called the Cookie Exchange (note that the term "cookie" is used differently than in the context of the World Wide Web and the corresponding HTTP state management protocol that is addressed in Section 4). The Cookie Exchange is briefly analyzed in Section 3, and corresponding design considerations for protecting key exchange and management protocols against resource clogging attacks are derived in Section 4. This section also elaborates on possibilities to use similar techniques to improve an existing HTTP state management protocol and to protect TCP implementations against SYN flooding attacks. Finally, conclusions are drawn in Section 5.

2. COOKIE EXCHANGE

In the early stages of the work of the IETF IPsec WG, Phil Karn developed and proposed a Photuris Session Key Management Protocol that was later submitted for possible standardization as IKMP. Note that in spite of the fact that a more recent version of the Photuris Session Key Management Protocol specification was published in February 1998 [KS98], this paper refers to a former (and outdated) version of the protocol specification (the one that was pusblished in November 1995). This poses no problem, since the paper does not focus on the specific characteristics of the Photuris Session Key Management Protocol, but rather on its Cookie Exchange mechanism, which has remained essentially the same throughout all subsequent versions of the protocol specification (and which has also remained the same for the IKE protocol specification). Also note that the protocol name is not an acronym, but rather a tribute to some unknown engineers. In fact, "Photuris" is the Greek name used by zoologists to designate the firefly, and "Firefly," in turn, is the name of a classified key exchange protocol designed by the U.S. National Security Agency (NSA) for the STU-III secure telephone. A rumor tells that the design of Photuris is very closely related to that of the Firefly protocol. The Photuris protocol is also conceptually similar to the Station-to-Station (STS) protocol originally proposed by Whitfield Diffie, Paul van Oorschot, and Michael Wiener [DOW92]. Some of the techniques used in the Photuris protocol are covered by U.S. Patent 5,148,479, granted to IBM. In August 1995, however, IBM announced that it would grant the free use of the patented technologies in conjunction with the Photuris protocol and its derivates to the IETF [Low95].

Similar to the Firefly and STS protocols, the Photuris Session Key Management Protocol combines a Diffie-Hellman (DH) key exchange with a subsequent authentication step for the public parameters used in the DH key exchange. The authentication is done with RSA signatures. In addition to the Firefly and STS protocols, the original Photuris protocol specification also incorporated an anti-clogging mechanism that was intended to be performed prior to the DH key exchange. The aim of this mechanism is to protect Photuris entities against simple flooding and resource clogging attacks with randomly chosen and bogus IP source addresses and UDP port numbers (note that the Photuris Session Key Management Protocol is layered on top of the connectionless User Datagram Protocol).

In spite of the fact that the Photuris protocol was not approved as IKMP or IKE protocol, its anti-clogging mechanism has been included into several other key exchange and management protocols, including the ISAKMP and the OAKLEY Key Determination Protocol. Consequently, the Photuris anti-clogging mechanism is part of the IKE protocol specification that is now being standardized. Since the Photuris anti-clogging mechanism includes an initial message exchange that is used to exchange anti-clogging tokens (ACTs) called "cookies," it is also called the Cookie Exchange.

The entities involved in a Photuris protocol execution are called initiator on the requesting side, and responder on the serving side. In short, the protocol execution comprises three phases (refer to [Opp98b] for a more comprehensive overview about the Photuris protocol):

- A Cookie Exchange;
- A Value Exchange;
- An Identification Exchange.

Before the Photuris initiator and responder enter the Value Exchange phase and perform a computationally expensive DH key exchange (which includes modular exponentiation), they perform the Cookie Exchange, in which the initiator sends a Cookie_Request message to the responder, and the responder returns a Cookie_Response message to the initiator. The two messages include Initiator-Cookie and Responder-Cookie fields, both providing room for 16-byte (or 128-bit) ACTs or cookies that look like random values to an outsider. The fixed size of a cookie was chosen for convenience, based on the output of some commonly available one-way hash functions, such as MD5 or RIPEMD-128 [MOV97]. Alternatively, the size of the Initiator-Cookie and Responder-Cookie fields could also be increased to 20 bytes (or 160 bits) for other one-way hash functions, such as the Secure Hash Algorithm (SHA-1) or RIPEMD-160 according to ISO/IEC 10118-3 [MOV97]. Recent results in cryptanalytic research have shown that it is even possible to truncate keyed one-way hash function results if the length of the corresponding cookies is critical and must be minimized [BCK96]. Making use of this possibility provides a way to make the mechanism more efficient.

More specifically, the IPsec Cookie Exchange works as follows: First, the initiator initializes some local state, and sends a Cookie_Request message to the responder. The initiator also starts a retransmission timer. If no Cookie_Response message is obtained within a certain amount of time, the Cookie_Request message is retransmitted. The Cookie_Request message includes an Initiator-Cookie field that contains a randomized value (notably the initiator cookie) to identify the exchange. The Initiator-Cookie field value in each retransmission to the same IP destination address and UDP port number (if no Cookie_Response message is received within the timelimit indicated by the retransmission timer) will actually be the same. For every new initiation of the protocol, however, the value of the Initiator-Cookie field will be different. The initiator uses this value to reject invalid responses. The Responder-Cookie field is zero or identifies a specific previous exchange (copied from a previous Cookie_Response message).

On receipt of the Cookie_Request message, the responder determines if there are sufficient resources to begin another Photuris key exchange. When too many security associations (SAs) and corresponding security parameters index (SPI) values are already in use for the requesting initiator, or some other resource limit is reached, a corresponding error message is returned. Otherwise, the responder generates a cookie, and returns it in the Responder-Cookie field of a Cookie_Response message (the Initiator-Cookie field of the Cookie_Response message contains the cookie that the initiator sent to the responder as part of the Cookie_Request message). Unlike the Initiator-Cookie field value, the Responder-Cookie field value may be different in each successive response. Note that the responder creates no additional state at this point in time. In particular, the responder does not store the value that is included in the Responder-Cookie field of the Cookie_Response message.

On receipt of the Cookie_Response message, the initiator validates the Initiator-Cookie field value. If the value is valid (meaning that it is identical to the one originally sent to the responder), the protocol execution continues with the Photuris Value Exchange and Identification Exchange phases. Otherwise, the message is silently discarded. In all subsequent messages the cookies are included in the Initiator-Cookie and Responder-Cookie fields and verified accordingly.

The exact technique by which a Photuris entity generates a cookie (either an initiator or responder cookie) is implementation dependent.

The original Photuris protocol specification only stated that the method chosen must satisfy the following three requirements:

1. The cookie must depend on the participating entities;
2. It must not be possible for anyone other than the issuing entity to generate a cookie that will be accepted by that entity;
3. The cookie generation and verification methods must be computationally efficient.

The first requirement prevents an attacker from obtaining a valid cookie using his real IP address and UDP port number, and then using the cookie to attack the responder with a sequence of Exchange_Request messages with randomly chosen and bogus IP source addresses and UDP port numbers. Making a cookie dependent on the participating entities is actually the most important requirement for an effective anti-clogging mechanism.

The second requirement implies that the generation and subsequent verification of a cookie must be cryptographically sound, and that the issuing entity must use some secret information in these computations. Without this secret information, an attacker must not be able to generate or verify a cookie. Also, he must not be able to deduce the secret information from a cookie. From a technical point of view, the situation is rather simple, since the secret information is local and must not be distributed to peer entities. Cookie generation and verification is a local matter that does not involve any remote peer entity; this simplifies the management of secret information considerably.

Finally, the third requirement is to thwart resource clogging attacks against the anti-clogging mechanism itself. Note that if an anti-clogging mechanism involved some computationally expensive operations, it would also be possible to launch a resource clogging attack against this particular mechanism (instead of launching the attack against the key exchange or management protocol that is protected with the mechanism). Consequently, an effective anti-clogging mechanism must use efficient cryptographic algorithms and techniques, such as keyed one-way hash functions [Gon89,Tsu92].

According to the original Photuris protocol specification, a recommended technique (to generate and verify a cookie) is to compute a one-way hash function, such as MD5, over the IP source and destination addresses, the UDP source and destination port numbers, keyed with some locally generated secret value (which must not be different for different security associations). Any construction for keyed one-way hash functions can be used to generate and verify the cookies, the HMAC construction being a good candidate [BCK96,KBC97]. Note that the use of the secret value and the corresponding computation of the keyed one-way hash function is slightly different on the initiator and responder sides.

- On the initiator side, the secret value that affects the cookie should change for each responder; it is thereafter internally cached on a per responder basis. This provides improved synchronization and protection against replay attacks. An alternative is to cache the cookie instead of the secret value. Incoming cookies can then be compared directly without the computational costs of cookie regeneration.

- On the responder side, the secret value may remain the same for many different initiators. Nevertheless, the latest version of the Photuris protocol specification requires a periodical change of the secret value and suggests changing it every 60 seconds. During the Value and Identification Exchanges, the responder regenerates the corresponding cookie for each validation. The responder cookie is not cached during the initial Cookie Exchange to avoid save state. Only after the first message of the Value Exchange phase (namely the Exchange_Request message) is received, both the initiator and responder cookies are cached to actually identify the exchange.

The important point to note is that the responder doesn't create any state prior to receiving a valid cookie from the initiator to which it has previously issued it. The cookie is validated by comparing it to a reconstructed value based on information in the incoming Exchange_Request message (the first message in the Photuris Value Exchange phase) plus the appropriate secret information. It does take some time to run the keyed one-way hash function, but it is minimal. It is certainly better than creating state and running the risk of being swamped in an attack similar to that which might be mounted with a sequence of TCP SYN messages [SKK+97].

3. BRIEF ANALYSIS

In this section, we briefly analyze the security properties of the Photuris or IPsec Cookie Exchange. Assuming the method to construct cookies with a keyed one-way hash function (e.g., using the HMAC construction) to be cryptographically sound, an attacker has the following possibilities to circumvent the anti-clogging mechanism provided by the Cookie Exchange:

- First, the attacker can request a cookie with his true IP address and UDP port number, and use the returned cookie to launch a resource clogging attack;

- Secondly, the attacker can request a cookie with a wrong IP address and UDP port number, and use the returned cookie to actually launch the resource clogging attack;

- Thirdly, the attacker can also eavesdrop on a communications line and grab some valid cookies that are passing by in order to launch the resource clogging attack.

The first attack is yet possible but reveals the true IP address of the attacking entity (disabling the possibility to stay anonymous). Consequently, this attack is not considered as a serious threat and can be countered accordingly.

Unlike the first attack, the attacking entity can stay anonymous in the second attack. Note, however, that this attack is somehow more difficult to launch, since the attacked entity will return the responder cookie to the wrong IP address. The attacker must therefore either grab the response message on its route back to this (the wrong) IP address or enforce a route that passes through his site. A possibility to enforce this route may be the use of the IP source routing option in the cookie requesting message. A more pragmatic attacker would try to compromise a system that is phyically located on the same network as the victim and request the cookies from there. Since the compromised system is located on the same network it can also be used for grabbing the returned cookie and forwarding it to the attacker.

In either case, the most serious threat is due to the third attack in which the attacker uses valid cookies to launch a resource clogging attack (spoofing appropriate IP addresses and UDP port numbers). Again, an attacker can make use of a compromised system that is physically located on the same network as the victim to eavesdrop and collect valid cookies. Note, however, that according to the way in which cookies are generated, they are restricted to use from specific IP addresses and UDP port numbers. Consequently, the attacker must spoof these IP addresses and UDP port numbers in order to initiate key exchange protocol executions and launch a corresponding resource clogging attack. In the following section two design considerations are derived that can be used to minimize the risks that result from this threat.

4. DESIGN CONSIDERATIONS

In an authenticated key exchange protocol, the key exchange may be performed either before or after the authentication step:

- In the first case, a shared secret is established before authentication is performed. Consequently, the identities of the participating entities can be protected or hidden with the shared secret, and some form of anonymity service can therefore be provided. Unfortunately, an outsider is also able to launch a resource clogging attack under a wrong identity (since the entity is not authenticated prior to the key exchange).
- In the second case, authentication is performed before a shared secret is established. Consequently, anonymity services are difficult to provide and resource clogging attacks are hard to launch under a wrong identity (since the entity must be authenticated prior to the key exchange).

There is an interesting trade-off between the possibility to hide the identities of the participants of an authenticated key exchange and the possibilities of an outsider to anonymously launch a resource clogging attack. Consequently, protecting a key exchange or management protocol against resource clogging attacks is important if the following two conditions hold:

1. The protocol performs an authenticated key exchange in which the key exchange is performed prior to the authentication;
2. The protocol is layered on top of a connectionless protocol, such as the Internet Protocol (IP) on the network layer or the User Datagram Protocol (UDP) on the transport layer.

Most key exchange and management protocols that have been proposed and submitted for standardization to the IETF IPsec WG fulfill these requirements. This is equally true for many other key exchange and management protocols. Note, however, that some key exchange and management protocols that are in widespread use today, such as the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols [Opp98b], are layered on top of the Transport Control Protocol (TCP) that is connection-oriented. This situation is less vulnerable to resource clogging attacks, since a TCP connection must first be established before a victim can be flooded with key exchange request messages, such as ClientKeyExchange messages in the case of SSL/TLS (this follows a more general line of argumentation that connection-oriented protocols are generally simpler to secure).

The IPsec Cookie Exchange provides some protection against resource clogging attacks. Having the analysis of the last section in mind, there are two considerations that one may think of in order to improve the effectiveness of an anti-clogging mechanism:

1. *Restricting the usability of a cookie to just one simultaneous protocol execution:* In general, a cookie is requested by an initiator of a key exchange or management protocol execution and is used by this entity exclusively. Consequently, there is no need to allow a cookie to be used for several protocol executions simulatenously, and the use of such a cookie can be detected and aborted.

2. *Restricting the lifetime of a cookie:* In general, a cookie may have a restricted lifetime and an initiator who has received a cookie may have to use it within a given time interval. There are several possibilities that one may think of in order to restrict the lifetime of a cookie and to implement cookie aging accordingly:

   - One possibility is to use timestamps and synchronized clocks. For example, the ISAKMP requires inclusion of a timestamp in the cookie generation and restricts its lifetime to the lifetime of one security association (SA).
   - Another possibility would be to periodically change the secret information that is used to generate and verify the cookies.

An additional word is due to the use of the mechanisms and technologies addressed in this paper to protect a key exchange or management protocol against resource clogging attacks to improve the security properties of "cookies" as used in a widely deployed HTTP state management protocol [KM97]. In essence, the use of such cookies allows an HTTP server to create a stateful session with specific HTTP request and response messages. In general, cookies are downloaded and stored on the client side without protection. Anybody with write access to the corresponding file (which is `cookies.txt` in a Netscape environment) can either create or modify and resubmit a cookie without giving the server a chance to realize the change. Consequently, a data origin authentication and integrity service would be useful to allow a server to recognize whether a cookie has been modified since its original release. Again, this can be done using a keyed one-way hash function to compute a message authentication code (MAC) for the cookie, and again, the secret value that is used to key the one-way hash function must be known to the server only. This simplifies key management considerably.

Finally, note that a similar anti-clogging mechanism can also be used to protect a TCP implementation against SYN flooding attacks [SKK+97]. In short, a TCP SYN flooding attack works as follows: When a TCP entity receives a SYN message, it creates some local state, returns a SYN-ACK message, and waits for the requesting entity to finish the connection establishment with a final ACK message. During this waiting period, some buffer space is occupied (at least until a timer expires), and since the entire buffer space is limited, a TCP entity can be flooded quite easily through a sequence of TCP SYN messages. In oder to stay anonymous, an attacker typically sends out these messages with random IP source addresses.

An obvious method to counter the TCP SYN flooding attack is to increase the size of the buffer space. Note, however, that this does not solve the problem entirely, but only makes the attack more difficult to launch (since it only requires more TCP SYN messages to be sent by the attacker).

In order to effectively counter the TCP SYN flooding attack (launched with random IP source addresses), the creation of local state must be avoided on the server side until it can be verified that the requesting entity is actually using its true IP address. This can be done in a way that is similar to the Photuris or IPsec Cookie Exchange: Let's assume a TCP entity S (standing for server) receives a SYN message from another TCP entity C (standing for client) with a sequence number field value set to X. Let's further assume that S does not create any local state at this point in time, but directly returns a SYN-ACK message instead. In this message, the acknowledgment number field value is set to $X + 1$ (indicating the next sequence number that it expects) and the sequence number field value is set to Y that represents a keyed one-way hash function result truncated to 32 bit. Again, the message for which the keyed one-way hash value is generated includes the IP addresses and TCP port numbers of C and S (as well as some other information), and again, the secret value that is used to key the one-way hash function is a local matter and must not be distributed in one way or another. When S finally received a TCP ACK message that includes an acknowledgment number field value of $Y + 1$, it would verify the validity of $Y + 1$ according to the information found in the corresponding message headers (IP addresses and TCP port numbers). If the value were valid, local state would be built for the TCP connection with C. Otherwise, the connection establishment request would be discarded and no state would actually be built. Note that the use of a keyed one-way hash function to compute the sequence number field value has better random characteristics than generally used methods, making IP spoofing attacks generally more difficult to launch.

Obviously, the analysis for the IPsec Cookie Exchange also applies for this mechanism to counter the TCP SYN flooding attack. Note, however, that the sequence number field in the TCP header is only 32 bits in length, and that the keyed one-way hash function result must be truncated to this length accordingly. This may give some concerns, since a one-way hash function producing a 32-bit value is generally far away from being collision-resistant (an attacker can find a message that hashes to a certain value in 2^{31} trials on the average). Note, however, that an attacker does not know the secret value that is used to key the one-way hash function. If this key is sufficiently large, the attacker will not be able to find any collision that he may (mis)use in oder to correctly spoof his IP address and TCP port number.

5. CONCLUSIONS

Many cryptographic key exchange and management protocols involve computationally expensive operations, such as modular exponentiations, and are therefore vulnerable to resource clogging attacks. This paper has overviewed and discussed the basic principles and the rationale behind an anti-clogging mechanism that was originally designed and proposed to protect the Photuris Session Key Management Protocol against resource clogging attacks. In spite of the fact that the Photuris protocol was not approved as Internet Key Management Protocol (IKMP) or Internet Key Exchange (IKE) protocol respectively, the current IKE protocol specification inherits its anti-clogging mechanism. Based on a brief analysis of this mechanism, the paper has also derived some design considerations for anti-clogging mechanisms and has elaborated on possibilities to use similar techniques to improve an existing HTTP state management protocol and to protect TCP implementations against SYN flooding attacks.

As protection against denial-of-service and degradation-of-service attacks is becoming more and more important for Internet-based applications, further research is expected to be done in this area. For example, the use of anti-clogging mechanisms in cryptographic (and non-cryptographic) protocols deserves further study, and the trade-off between the possibility to hide the identities of the participants of an authenticated key exchange and the possibilities of an outsider to anonymously launch a resource clogging attack has also not been studied in its entirety. Finally, prototype implementations will have to be done to get some evidence about the performance characteristics of anti-clogging mechanisms.

Acknowledgments

The author thanks Phil Karn for explaining the rationale behind the Photuris Cookie Exchange and for sharing his ideas regarding the use of similar techniques to protect a TCP implementation against SYN flooding attacks. In addition, the author also thanks the anonymous reviewers for their useful comments.

References

[Atk97] R.J. Atkinson. Toward a More Secure Internet. *IEEE Computer*, January 1997, pp. 57 - 61

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keyed Hash Functions and Message Authentication. Proceedings of CRYPTO '96, pp. 1 - 15

[DOW92] W. Diffie, P.C. van Oorshot, and M.J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 1992, pp. 107 - 125

[Gon89] L. Gong. Using One-Way Functions for Authentication. *ACM Computer Communication Review*, Vol. 19, No. 5, October 1989, pp. 8 - 11

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, February 1997

[KM97] D. Kristol and L. Montulli. HTTP State Management Mechanism. RFC 2109, February 1997

[KS98] P. Karn, and W.A. Simpson. Photuris: Session-Key Management Protocol. Internet Draft, February 1998, work in progress

[Low95] J. Lowe. A Grant of Rights to Use a Specific IBM patent with Photuris. RFC 1822, August 1995

[MOV97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, FL, 1997

[Opp98a] R. Oppliger. Security at the Internet Layer. *IEEE Computer*, September 1998

[Opp98b] R. Oppliger. *Internet and Intranet Security.* Artech House Publishers, Norwood, MA, 1998

[SKK$^+$97] C.L. Schuba, I.V. Krsul, M.G. Kuhn, E.H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a Denial of Service Attack on TCP. Proceedings of IEEE Symposium on Security and Privacy, May 1997, pp. 208 - 223

[Tsu92] G. Tsudik. Message Authentication with One-Way Hash Functions. *ACM Computer Communication Review*, Vol. 22, No. 5, October 1992, pp. 29 - 38