Denial of Service Resistance in Key Establishment

J. Smith, S. Tritilanunt, C. Boyd, J. M. González Nieto, and E. Foo
Information Security Institute
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001
Australia

Abstract:
Denial of service (DoS) attacks are an increasing problem for network connected systems. Key establishment protocols are applications that are particularly vulnerable to DoS attack as they are typically required to perform computationally expensive cryptographic operations in order to authenticate the protocol initiator, and to generate the cryptographic keying material that will subsequently be used to secure the communications between initiator and responder. The goal of denial of service resistance in key establishment protocols is to ensure that attackers cannot prevent a legitimate initiator and responder deriving cryptographic keys without expending resources beyond a responder determined threshold. In this work we review the strategies and techniques used to improve resistance to denial of service attacks. Three key establishment protocols implementing denial of service resistance techniques are critically reviewed and the impact of misapplication of the techniques on denial of service resistance is discussed. Recommendations on effectively applying resistance techniques to key establishment protocols are made.

Keywords: Denial of service, cryptographic protocols, key establishment, key agreement, proofs of work, client puzzles.

Reference to this paper should be made as follows: J. Smith, S. Tritilanunt, C. Boyd, J. M. González Nieto, and E. Foo (2006) "Denial of Service Resistance in Key Establishment", *International Journal of Wireless and Mobile Computing*

Biographical notes: Jason Smith is currently a research associate with the Information Security Institute at Queensland University of Technology (QUT) and is working towards his PhD. His research interests include mobile and wireless network security and intrusion detection, cryptographic protocols, and denial of service resistance.

Suratose Tritilanunt received a Master Degree in Computer Engineering from King Mongkut's University of Technology Thonburi (KMUTT), Thailand. He is currently a PhD student in the Information Security Institute, QUT. His research interests include Cryptographic protocol, Denial-of-Service attack, Key Establishment protocol, Formal Methods, and Network Security.

Colin Boyd is a Professor in the School of Software Engineering and Data Communications and Deputy Director of the Information Security Insitute at QUT. His research interests are in the theory and applications of cryptography, particularly cryptographic protocols. He is the author of over 100 fully refereed publications including a prominent recent book on protocols for authentication and key establishment.

Juan González Nieto is a Research Fellow at the Information Security Institute, QUT. He obtained his PhD in the field of cryptographic protocols. His main research interests lie in the area key establishment protocols and public key cryptography, with an emphasis on provable security. He has authored over 19 journal and conference papers.

Ernest Foo is a lecturer in the School of Software Engineering and Data Communications at QUT and also a member of the Information Security Institute. The main topics of his research include secure protocols and network security with particular emphasis on electronic commerce applications.

1 INTRODUCTION

Key establishment protocols enable two parties to authenticate each other and establish cryptographic key material using an adversarially controlled network. The cryptographic keys derived from the key establishment protocol are then typically used to secure the communications channel by providing confidentiality and integrity services using IPsec [1] for example. The party that initiates the key establishment protocol by transmitting the first message is known as the initiator. The party that receives and responds to the first message is known as the responder.

In addition to providing a mechanism for establishing cryptographic keys, key establishment protocols may also provide additional services including: identity protection for the initiator or responder; plausible deniability, a property that ensures that evidence of participation by a particular party in a protocol run is not generated by the protocol; and perfect forward secrecy, a property that ensures compromise of long term keys cannot reveal previously negotiated session keys.

The need to secure the confidentiality and integrity of communications highlights the threat from malicious entities active in the network. Increasingly, malicious network entities are seeking not only to violate the confidentiality and integrity of information, but also to deny or degrade access to network services via denial of service (DoS) attacks.

Denial of service attacks can be classified as flooding, or logical (non-flooding). In flooding-based denial of service attacks, the attacker generates spurious network traffic or requests for service in order to exhaust available server resources, thereby preventing access to the service by legitimate clients. Directing a continuous stream of traffic that exceeds the bandwidth available to a target service is an example of a flooding attack. To successfully mount a flooding-based attack the attacker must command resources (processing, memory or network bandwidth) in excess of those provisioned at the target system. While the resources required for an attacker to successfully mount a flooding-based denial of service may be considerable, this type of attack requires limited knowledge of the specific service protocols or their implementation to be successful.

In logical attacks, rather than blindly trying to exhaust server resources, the attacker exploits characteristics or vulnerabilities of network protocols or applications in such a way as to exhaust available resources. Exploiting vulnerabilities in the target system[1] or protocols employed by the target system, such as with the TCP based SYN flooding attacks [3] are examples of logical attacks. Logical attacks require the attacker to have a greater understanding of the protocols or applications to be targeted, but can dramatically reduce the resources required to successfully attack a service.

Needham [4] identified that attacks may be directed at: (1) servers, henceforth referred to as responders; (2) the network infrastructure, such as links, routers, domain name servers; or (3) specific client systems, henceforth referred to as initiators. Having determined where an attack is to be directed, an attacker can attempt either a flooding or logical attack.

Key establishment protocols are applications that are particularly vulnerable to logical denial of service attacks as they are typically required to perform computationally expensive cryptographic operations in order to authenticate the protocol initiator, and to generate the cryptographic keying material that will subsequently be used to secure the communications between initiator and responder. Additionally, key establishment protocols are openly specified, providing attackers with the opportunity to easily search for denial of service vulnerabilities in the protocol specifications.

Assuming the presence of malicious network entities, key establishment protocols must adopt strategies and techniques to ensure that they are not susceptible to denial of service attacks. The ability of a key establishment protocol to withstand attempts to exhaust responder resources via denial of service attack is termed the *denial of service resistance* of the protocol. The goal of denial of service resistance in key establishment protocols is to ensure that attackers cannot prevent a legitimate initiator and responder deriving cryptographic keys without expending resources beyond a responder determined threshold.

While we recognize that to be effectively managed denial of service must be addressed at the link, network, operating system, and application layers, we focus our attention in this work specifically on the denial of service resistance techniques employed at the application layer by key establishment protocols. While either the initiator or responder may be the target of a denial of service attack, in this paper we only consider malicious initiators and the techniques used to defend responders against attack. The reason for focusing on responders is that responders typically handle requests for service from many initiators, so the impact of denial of service attacks against a responder is significant. The effect of a denial of service attack targeted at an initiator is far more localised, only impacting the specific target of the attack.

Even though numerous strategies and techniques are available to improve the resistance a protocol has to denial of service attack, they are not always applied correctly resulting in protocols that do not effectively improve their resistance to denial of service attack. The contributions of this work include:

- a review of the strategies and techniques used to improve resistance to denial of service attack;
- the critical analysis of protocols implementing denial of service resistance techniques with discussion of the impact of misapplication of the techniques on denial of service resistance; and

[1] Errors in operating system handling of oversized ICMP echo requests led to some systems to crash, freeze or reboot on receipt of a "ping of death" [2].

- recommendations on effectively applying denial of service resistance techniques to key establishment protocols.

Owing to space constraints, we only present a subset of protocols that can serve as exemplars for the techniques identified.

The remainder of the paper is structured as follows: Section 2 identifies the strategies used to increase resistance to denial of service attack; Section 3 presents the techniques used to implement denial of service resistance strategies; Section 4 details three protocols that implement denial of service resistance strategies, critically discussing the application of the techniques; and Section 5 presents conclusions and directions for future work.

2 DOS-RESISTANCE STRATEGIES

A responder in a key establishment protocol will have access to finite processing, storage, and network resources in order to complete its functions. Unless these resources are committed diligently, they may be exhausted by malicious initiators and the responder will have insufficient resources remaining to process legitimate incoming requests.

The requirement for key establishment protocols to exhibit denial of service resistance is well recognized by the protocol engineering community and a number of design strategies have emerged that promote the judicious allocation of resources when processing initiator requests [5, 6]. The proposed strategies can be broadly classified into three types.

1. **Counterbalancing memory expenditure** By ensuring that initiators must commit their memory resources to maintaining protocol state until the responder has some assurance that a denial of service attack is not underway reduces the vulnerability of the responder to state or memory-based denial service attacks and increases the memory resources an attacker will need to attack the responder.

2. **Counterbalancing computational expenditure** By counterbalancing computational expenditure at the responder, the protocol designer can ensure that the computational resources of an initiator will be exhausted before those of the responder. Achieving this goal may require artificially increasing the computational expenditure of the initiator to ensure the survivability of the responder [5], or having the initiator perform computations on behalf of the responder, thereby reducing the relative cost of computation to the responder.

3. **Gradual authentication** While initiators must be authenticated at some point during the protocol execution, immediate and strong authentication of requests merely aggravates the denial of service problem. The suggested strategy for balancing the need for authentication and computational expenditure is to use weak and computationally cheap authentication when the protocol is initiated and gradually increase the strength of authentication as the protocol proceeds [6]. The strategy of gradual authentication can be used to detect attacks (based on IP spoofing for example), verify the computational and memory commitments of the initiator, and link messages from the same source (even though the exact identity of that source may be unknown).

   The protocols discussed in Section 4 provide examples of how gradual authentication can be implemented in key establishment protocols.

Combining the strategies of counterbalancing computational expenditure, counterbalancing memory expenditure, and gradually authenticating requests ensures that malicious initiators are unable to prevent the establishment of cryptographic keys between legitimate initiators and responders, unless they are prepared to expend significant resources of their own.

3 DOS-RESISTANCE TECHNIQUES

Having identified the strategies employed to make responders in a key establishment protocol more resistant to denial of service attacks in Section 2, we now describe the specific techniques used. The techniques described may be considered primitives, some of which are capable of implementing more than one strategy and some of which can be combined to meet more complex goals such as gradual authentication. For each technique identified, we discuss its construction, the DoS resistance strategies it is capable of supporting, and how it might be combined with other techniques. Protocols implementing the techniques are critically discussed in Section 4.

3.1 Cookies

Cookies are time variant, unpredictable data issued by the responder on receipt of a request for service that allow the responder to remain stateless and initiate gradual authentication of the initiator. First introduced in Photuris [7] and subsequently extended for resisting SYN flooding DoS attacks [8], cookies are now widely used.

Typically a cookie is constructed by taking some connection specific parameters and transforming them with a time variant local secret; a keyed hash of the initiator IP address and nonce for example. It is vitally important that the responder store no state when constructing cookies. In order to remain stateless and thereby prevent memory exhaustion, any relevant state required by the responder can also be encoded in the cookie and returned with the next message from the initiator. An approach for making protocols stateless is presented by Aura and Nikander [9].

On receipt of a valid cookie, the responder is able to reconstruct and validate any state encoded in the cookie and has weak assurance that it is in round trip communication with the initiator. Round trip communication implies that the initiator is not using a spoofed address. This assurance can only be considered weak, as an adversary with control of an intermediary link, between a claimed address and the responder, would be able to receive cookies for any address they wished to claim.

Unless cookies are carefully constructed the responder may remain vulnerable to attack even if cookies are used. Simpson [10] identified a state exhaustion attack, called a "cookie crumb" attack, in the ISAKMP implementation of cookies. In contrast to remaining stateless when constructing cookies, ISAKMP cookies required the storage of a small amount of state on each connection request. Even though the state information stored per request is very small (a "crumb") it is easy for an attacker to initiate a large number of requests, exhausting available memory resources.

In addition to ensuring that no state is stored on the construction of a cookie, Karn and Simpson [7] identified that the technique used for generating cookies must also satisfy the following three requirements.

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

The first requirement prevents an attacker from obtaining valid cookies, intended for other initiators, and using those cookies to generate a large number of requests with spoofed IP addresses. The second requirement secures the cookie generating process. The use of a secret value in generating the cookie prevents others from forging cookies and making this value time variant ensures that cookies must be used within a predetermined time frame, preventing the hoarding of valid cookies. Finally, the third requirement prevents DoS attacks directed at the cookie mechanism itself.

3.2 Proofs of Work

Proofs of work, or puzzles, are hard but tractable problems that allow an initiator to prove to a responder that a verifiable level of computational effort has been expended. They permit the responder to gain some assurance of the initiator's willingness to commit resources to the protocol and provide a mechanism for counterbalancing computational expenditure in the event that the responder is exposed to a denial of service attack.

The concept was first proposed by Dwork and Naor [11] to control junk email by having recipients only accept emails if they were accompanied by a correct puzzle solution. It has since been extended to protect authentication protocols [12, 13] and permit clients to bid for limited service resources [14] using the difficulty of the puzzle as currency. Jakobsson and Juels [15] formalised the notion of reusable proofs of work, where the computational effort expended by the prover in generating the puzzle solution can be reused for some useful function, and provided a working example of a reusable proof of work.

Puzzles serving as proofs of work can be constructed from a number of underlying problems, which introduce a minimal and configurable overhead for legitimate initiators but result in a significant computational burden for attackers who wish to send large numbers of requests to a responder.

Properties of a good puzzle include [12, 13]:

- generation and verification is inexpensive for the responder;
- level of difficulty can easily be adjusted from trivial to impossible;
- solutions should not require specialised client hardware;
- solutions cannot be precomputed;
- issuing a puzzle does not require the responder to store any state;
- knowledge of the solution to one client's puzzle is of no benefit in solving other puzzles, so that the same puzzle may be provided to numerous clients; and
- initiators can reuse a puzzle by creating new instances of it.

3.2.1 Hash-based puzzles

Juels and Brainard [12] describe the construction of client puzzles to protect TCP and SSL against connection depletion (SYN flooding) attacks. In their proposal, when a server becomes heavily loaded (inferred from buffer occupancy), connections are only accepted if they are accompanied by a proof of work. The puzzle (shown in Figure 1(a)) is constructed by hashing session parameters (M), the current time (t), and a responder secret (s). The n-bit output of this hash operation (X) becomes the preimage to another application of a hash function, whose output (Y) forms part of the puzzle. The initiator is provided the partial preimage X' (X with k-bits masked out), and the hash digest Y. In order to solve the puzzle, the initiator must test all k possible preimages until the correct output is achieved. On average this will take 2^{k-1} hash operations.

$$\begin{aligned} hash(t||M||s) &= X \\ hash(X) &= Y \\ X' &= X \ \& \ 0_0, 0_1, ..., 0_{k-1} 1_k 1_{k+1} 1_{n-1} \\ puzzle &= (X', Y) \\ solution &= p \\ verification &: hash(p||X') \overset{?}{=} hash(t||M||s) \end{aligned}$$

(a) Juels and Brainard Construction

$$hash(N_R||X) = \underbrace{000, ..., 000}_{k-bits} ||, ...$$

(b) Aura et al. Construction

Figure 1: Hash-Based Puzzle Constructions

To verify the solution (p), the responder checks the time value t is recent, and then confirms, with a single hash operation, that the proposed solution is correct:

$$hash(p||X') \overset{?}{=} hash(t, M, s)$$

An alternative construction is proposed by Aura et al. [13] and is shown in Figure 1(b). In this proposal the puzzle consists of a time variant responder nonce (N_R) and a difficulty parameter (k). To solve the puzzle the initiator must find the value X, that when hashed with the responder nonce (N_R) produces a digest output whose first k-bits are zeros.

The responder verifies the puzzle solution by checking that N_R is recent and that the solution (X) when hashed with the nonce produces an output with the first k-bits as zero.

Signed puzzles The construction of Aura et al. [13] was designed specifically for use in authentication protocols. The construction does not include any initiator specific parameters, allowing the responder to sign a single puzzle and issue it to multiple initiators. Conversely, the inclusion of initiator specific parameters in the Juels and Brainard [12] construction would make the signing of puzzles prohibitively expensive.

Puzzles, reachability and statelessness To operate effectively as a replacement for cookies (providing weak authentication of initiator reachability and stateless connections) proofs of work must be constructed so that:

- the IP address of the initiator must be part of the puzzle construction, and
- required state information is also encoded in the puzzle.

If the puzzle construction is unable to meet these requirements, then the puzzle cannot replace the functions of a cookie in the protocol. If we reconsider the puzzle construction of Aura et al. [13], which does not include the IP address of the initiator in the puzzle construction, it would be possible for a malicious initiator to receive a puzzle challenge on one IP address and construct puzzle solutions for any number of spoofed IP addresses. Admittedly, the initiator would have to generate a unique solution for each address it wished to claim, but in spite of receiving a valid puzzle solution, the responder would have no assurance that the initiator was actually reachable at the claimed IP address. In order to gain assurance that an initiator is able to send and receive messages from a claimed IP address, protocols using the Aura et al. puzzle construction as a proof of work should also use a cookie. The Juels and Brainard [12] puzzle includes the IP address of the initiator in its construction, so protocols using this construction are able to use the puzzle as a replacement for a cookie.

3.2.2 Other constructions

While the hash-based construction is prevalent in interactive protocols owing to its simple construction and cheap verification, other puzzle constructions have been proposed to accommodate: non-interactive protocols such as email; puzzles that are intended to act as time capsules with solutions taking years, not milliseconds; and techniques for supporting the outsourcing of puzzle constructions.

- **Signature-based puzzles**: The puzzle proposed Dwork and Naor [11] could be constructed from a weakened Fiat-Shamir signature, or from the broken Ong-Schnorr-Shamir signature scheme. The initiator is required to forge a verifiable signature as a proof of work to the responder. Matsuura and Imai [16] present an alternative form of signature-based proof of work in which verification of the responder signature by the initiator requires the calculation of an intermediate value. This value is presented by the initiator to the responder as proof that the signature verification, and the associated modular exponentiations, have been performed (See Section 4.1 for the details).

- **Time-Lock puzzles**: Proposed by Rivest et al. [17], time-lock puzzles are based on the notion that a client has to spend a predetermined amount of computation time performing repeated squaring to find a solution. The server calculates the number of squaring operations which a client can perform, and determines the amount of time it wants a client to spend solving the puzzle. Time-lock puzzles are inherently sequential and non-parallelisable.

- **Diffie-Hellman puzzles**: Waters et al. [18] investigate numerous techniques for outsourcing the generation of puzzles in order to remove the computational burden of puzzle generation from the responder. One of their proposals is based on a Diffie-Hellman construction, in which given a generator g, and a random value a in the range r to $r+k$, the puzzle issued to the initiator contains the values g^a and r. The initiator searches for a solution by trying each candidate value in the range r to $r+k$ until it finds c, such that $g^c = g^a$. To bind the solution to a specific server, the initiator calculates y^a, where y is the responder's Diffie-Hellman public key. The responder verifies the solution with a single modular exponentiation, raising the challenge value to the exponent (x) of its private key i.e. g^{ax}. The system described by Waters et al. [18] limits puzzles to a given timeslot. The solutions for a given timeslot are all precomputed by the responder, so puzzle verification is performed by a table lookup - not an online modular exponentiation.

Proofs of work can be viewed as a way for an initiator to make a payment to a responder for the services it will provide. The computational effort expended in generating the proof of work can be: (1) wasted; (2) reused by the initiator in completing the protocol; (3) reused by the initiator for some other purpose; (4) reused by the responder in completing the protocol; or (5) reused by the responder for some other purpose.

The client puzzles employed by the protocols identified in the following section, are typically non-reusable proofs of work, so the computational effort expended in generating the proof of work is wasted. We will see however, that the modified version of Internet Key Exchange (IKE) proposed by Matsuura and Imai [16] adopts a reusable proof of work based on signature verification of the responder, an action that a legitimate initiator will have to perform in order to complete the protocol.

As identified earlier, Jakobsson and Juels [15] describe a proof of work in which the computational effort expended in generating the proof of work is reused by the responder for another application. There are currently no examples of key establishment protocols implementing this type of reusable proof of work and unless an initiator implicitly trusts the responder to delegate its computational resources, initiators must be aware that the computational effort expended in generating a proof of work may be reused for malicious purposes.

Even though proofs of work are increasingly being adopted by protocols to aid in the counterbalancing computational expenditure and as a way of authenticating the willingness of an initiator to commit resources to having the protocol proceed, their use still faces numerous issues.

- Hash-based constructions meet many of the desirable properties of proofs of work (puzzles), but they also have the property that exhaustive searching of a preimage search space is a parallelisable task. Using such a technique in the presence of an adversary with access to distributed computing resources may leave the protocol exposed to denial of service. Adopting alternate puzzle constructions, such as time lock puzzles, that are inherently sequential and non-parallelisable may need to be considered for protocols that are to be used in an environment where the adversarial model assumes that significant resources are available to the attacker.

- Proofs of work based on processor bound functions result in puzzle constructions that can be solved in negligible time on a modern desktop computer, but may take an inordinately long time on light weight devices such as mobile phones or personal digital assistants. Given an effective strategy for authenticating the platform that a puzzle was being issued to, could allow the responder to tune the difficulty of the puzzle. As there is no way to authenticate the platform a puzzle is being issued to, alternative constructions based on memory bound functions are being investigated [19, 20]. Memory bound functions should exhibit a more uniform response time across a range of devices, as the difference in memory access speeds between light weight and more powerful devices is far less significant than the difference between processor speeds.

3.3 Client-Aided Computation

An alternative approach to artificially increasing computational expenditure at the initiator with puzzle constructions is to have the initiator perform computations that ease the computational burden of the responder. Client aided computations will most likely be performed before the initiator is fully authenticated, and as the initiator cannot be trusted at this time the type of computations that can be offloaded to the initiator are restricted to those that can be verified as correct.

Currently, the only protocol implementing client aided computation is the client aided RSA implementation of the SSL protocol proposed by Castellucia et al. [21] which is described in Section 4.3.

3.4 Gradual Authentication

While the expense of strongly authenticating initiators using digital signatures will be dependent on many parameters, the computational expense of a signature verification will not always be prohibitively expensive. Rabin signatures [22] with a public exponent of 2 or RSA signatures [23] with a public exponent of 3 can be verified with only one or two modular multiplications respectively. While the cost of signature verification with these parameters is low, signature generation is somewhat more expensive, which may not be suitable for all deployment scenarios. Other signature schemes, RSA with larger public exponents for example, increase the cost of signature verification, requiring the responder to perform expensive modular exponentiations. While newly proposed key establishment protocols can be specified to accommodate cheap signature verification for responders the requirement to improve resistance to denial of service attack remains for already deployed protocols and protocols, that for other reasons are restricted in the choice of signature schemes they must implement. Gradual authentication provides a mechanism for weakly authenticating an initiator, prior to performing stronger and more expensive cryptographic authentication.

The idea of combining weak and strong authentication was first introduced by Meadows [6] and is proposed as a technique to increase resistance to denial of service attacks by combining weak authentication when the protocol is initiated and moving to strong authentication as it completes.

Cookies and client puzzles can be considered forms of weak authenticators. Cookies provide some assurance that the initiator is able to send and receive packets from the claimed address–implying that the request is not part of a connection depletion attack, which typically relies on using random spoofed addresses. Receipt of a correct solution to a client puzzle provides some assurance to the responder that the initiator is willing to expend her own resources in order to get the protocol to proceed.

Other cryptographic techniques, such as the use of message authentication codes and release of hash digest preimages, that allow the responder to cheaply verify messages are being adopted by recently proposed protocols such as Just Fast Keying (JFK) as discussed in Section 4.

While the use of techniques such as cookies, client puzzles, and releasing hash preimages do not meet strong notions of authentication, when generated using cryptographically sound primitives they can be combined in ways which enable a responder to discount a range of denial of service attacks and present a number of hurdles that must be overcome by an attacker intent on disrupting the protocol execution. A key characteristic of the techniques used in gradual authentication is that they are all cheap for the responder to verify, while their fabrication is relatively expensive for an attacker. Even when signature schemes that minimise verification costs to a responder are adopted, the cost of verifying gradual authenticators such as client-puzzles is still cheaper, costing only a single hash operation.

Key establishment protocols must complete with strong authentication of the initiator. Having weakly authenticated the initiator, the responder is able to commit to the computational expenditure associated with strong authentication with increased assurance that a denial of service attack is not underway. The gradual authentication technique is employed by the modified Internet Key Exchange (IKE) and the Just Fast Keying (JFK) protocols, discussed in more detail in Section 4.

Table 1: Protocol Notation

Messages	Notation
I	The principal who initiates the request message known as Initiator or client
R	The principal who responds to the request message known as Responder or server
ID	Identity of the principal
IP	Network address of principal
$H(M)$	Unkeyed cryptographic hash of the message M
$H_K(M)$	Keyed cryptographic hash of the message M, with key K
$E_{K_s}\{M\}$	Symmetric encryption of message M with the secret key K_s
$\{M\}^{K_e}_{K_a}$	Encryption of M using symmetric key K_e, followed by MAC generation with symmetric key K_a.
$P[M]$	Asymmetric encryption of the message M by the public key P belonging to the principal
$S[M]$	Digital signature of message M with the private key S belonging to the principal
g	group generator of order q
N	Nonce of principal; a random bit string
K_R,s	The responder/server secrets
sa_I	Cryptographic and service properties of the security association (SA) that the initiator wants to establish
sa_R	SA information that the responder may need to give to the initiator
K_s	Session key generated by key establishment protocol which is used to secure ongoing communications
$grpinfo_R$	All groups supported by the responder
$x \in_R \mathcal{A}$	Assigns to x an element of the set $\mathcal{A}$ chosen uniformly at random
$N = PQ$	An RSA modulus

4 DOS RESISTANCE IN KEY ESTABLISHMENT

The process of authenticating initiators and generating cryptographic keys to secure ongoing communications requires responders to commit significant resources in order to have the protocol execute to completion. Unless key establishment protocols manage the commitment of their resources they will be susceptible to denial of service attacks.

In this section, we identify and discuss three protocols that implement strategies and techniques to improve their resistance to denial of service attacks. There are relatively few examples of key establishment protocols implementing denial of service resistance techniques, with our survey of the literature only identifying seven protocols (see Table 2 in Section 5 for a summary list). The three protocols presented in this section were selected as they provide concrete examples of the range of denial of service resistance techniques being applied to key establishment. The modified internet key exchange protocol proposed by Matsuura and Imai [16] adopts all three strategies for improving denial of service resistance and includes an elegant proof of work that is reused by the client to complete the protocol execution. The Just Fast Keying (JFK) protocol [24] demonstrates new techniques for gradually authenticating initiators, and the client-aided RSA proposal by Castelluccia et al. [21] is the first key establishment protocol to adopt client aided computation in order to counterbalance computational expenditure.

For the protocols presented in this section, we focus on those elements of the protocol that implement denial of service resistance techniques. Our representation of the protocols is simplified, with references to header information and certificate requests, that are not relevant to the discussion of denial of service resistance, deliberately omitted. For complete descriptions of the protocols, the reader is referred to the full protocol specifications.

We present the notation used for the remainder of this section in Table 1.

4.1 Modification of Internet Key Exchange Resistant against DoS

The Internet Key Exchange (IKE) protocol [25] was designed to perform mutual authentication and establish a shared secret key for use in an IPsec security association. As originally specified, the aggressive, signature based authentication mode of IKE was vulnerable to CPU and memory exhaustion denial of service attacks. In order to address these vulnerabilities Matsuura and Imai [16] proposed modifications to improve the protocol's resistance to both computational and memory based denial of service attacks. The modified protocol is presented in Figure 2 and adopts techniques that counterbalance computational and memory expenditure, and implement gradual authentication.

4.1.1 Counterbalancing memory expenditure

To address memory based denial of service attacks, this modified version of IKE stores no state after the first message. Unlike the original cookie construction that was vulnerable to a cookie crumb attack, the cookie in the modified proposal is constructed as a hash over request specific parameters, responder secret s, random fresh material a and Diffie-Hellman exponent r. The session specific secret parameters a and r are not stored by the responder, instead the protocol remains stateless by sending an encrypted copy of these parameters to the initiator.[2] In addition to allowing the responder to remain stateless, the cookie acts as a reachability test for the initiator, providing assurance that a spoofed address is not being used.

4.1.2 Counterbalancing computational expenditure

Computational denial of service attacks against the original aggressive mode of IKE with signature authentication resulted from the responder generating an expensive signature on the receipt of an unauthenticated message 1 and the expensive verification of a signature on message 3. To address these vulnerabilities Matsuura and Imai [16] specify the use of a signature scheme that permits expensive components of the signature generation to be precomputed and has a signature verification procedure that permits the recovery of random fresh material (a) used in the generation of the signature. The initiator then uses the recovery of the random fresh material a to provide proof to the responder that the signature in message 2 has been verified.

The expense of generating the signature sig_R is reduced but not eliminated in message 2 of the proposed IKE modification. The verification of the initiator signature (sig_I) is only performed after verifying that the initiator has incurred the computational expenses associated with verifying sig_R (as explained next).

In order to construct a message 3 that is accepted as genuine by the responder, the initiator must verify sig_R. Verification of sig_R, reveals the value a', which the initiator then uses to construct the proof of work $HASH^*_I$.

On receipt of message 3, the responder first validates the cookie and secondly verifies the proof of work by recovering a and verifying the modified initiator hash. Finally the responder can verify the signature of the initiator, with confidence that the initiator has already committed resources to the protocol execution.

While Matsuura and Imai's modified version of IKE [16] introduces an elegant proof of work based on verification of sig_R, it suffers from being an untunable parameter that provides no mechanism for the responder to increase the computational effort that must be expended by the initiator in order to provide the proof of work. Additionally, the technique is restricted to specific signature schemes with the Shortened DSS and the Schnorr signature schemes suggested by Matsuura and Imai [16] as capable of supporting precomputation and providing recovery of a.

4.1.3 Gradual authentication

In addition to counterbalancing computational and memory expenditure, this modified IKE allows the responder to gradually authenticate the initiator. Before verifying sig_I the responder has assurance that the initiator is not using a spoofed address, is willing to commit memory to having the protocol proceed, and has committed the computational resources required to verify the responder signature sig_R.

While the proposal represents an improvement over the original protocol, the responder is still required to commit computational resources to generate a signature on receipt of the first unauthenticated message leaving the responder susceptible to a flood of message ones.

4.2 Just Fast Keying

The Just Fast Keying (JFK) protocol was developed by Aiello et al. [24] as a key agreement protocol providing identity protection and capable of operating in a hostile environment such as the Internet. The protocol implements several techniques to counterbalance computational and memory expenditure, and to gradually authenticate initiator requests. The protocol variant implementing identity protection for the initiator is depicted in Figure 3.

[2] While not specifically indicated in the protocol specification, we suspect that not only the exponent, but the actual Diffie-Hellman value g^r must be securely sent to the initiator, otherwise the responder would have to recalculate the value on receipt of message 3.

	I		R
			$r, x_R \in_R [1, 2, \ldots, q-2]$
			$a = g^{x_R}$
1)	$i \in_R [1, 2, \ldots, q-2]$	$\xrightarrow{sa_I, g^i, N_I, ID_I}$	$Cookie = H(s, sa_R, IP_I, N_I, N_R, a, r)$
			$HASH_R = H(N_I \| N_R \| g^i \| g^r$
			$Cookie \| ID_R)$
			$s_2 = H(HASH_R \| a)$
			$s_1 = S_R \cdot s_2 + x_R \bmod q$
		$sa_R, g^r, N_R, ID_R,$	$sig_R = (s_1, s_2)$
2)	$HASH_R = H(N_I \| N_R \| g^i \| g^r$	$\xleftarrow{Cookie, E_{K_R}\{a \| r \| g^r\}, sig_R}$	
	$Cookie \| ID_R)$		
	$a' = g^{s_1} P_R^{-s_2}$		
	$s_2 \overset{?}{=} H(HASH_R \| a')$		
	$HASH_I^* = H(H(N_I, N_R),$		
	$g^i, g^r, Cookie, a', sa_R, ID_I)$		
	$K_s = H(N_I, N_R, g^{ir})$		
	$sig_I = S_I[HASH_I^*]$	$ID_I, N_I, HASH_I^*,$	
		$Cookie, E_{K_R}\{a \| r \| g^r\},$	
3)		$\xrightarrow{sa_R, g^r, N_R, sig_I}$	decrypt $E_{K_R}\{a \| r \| g^r\}$
			$Cookie \overset{?}{=} H(s, sa_R,$
			$IP_I, N_I, N_R, a, r)$
			$HASH_I^* \overset{?}{=} H(H(N_I, N_R),$
			$g^i, g^r, Cookie, a, sa_R, ID_I)$
			verify sig_I
			$K_s = H(N_I, N_R, g^{ir})$

Figure 2: Modified aggressive mode of IKE Protocol [16]

The responder periodically selects a Diffie-Hellman exponential (g^r) and generates a signature over this value and information on the groups it supports. The designers of the protocol allow the responder to reduce computational expenditure, at the expense of perfect forward secrecy, by reusing this Diffie-Hellman value with multiple initiators.

4.2.1 Counterbalancing memory expenditure

On receipt of the first message from an initiator, the responder remains stateless and weakly authenticates the reachability of the initiator by generating a cryptographic cookie ($Cookie = H_{HK_R}(g^r, N_R, N_I', IP_I)$), that is returned by the initiator in message 3. The secret key HK_R is a time variant local secret that limits the period of time a cookie will be accepted.

4.2.2 Counterbalancing computational expenditure

While the protocol permits reuse of the responder exponential to reduce computational expenditure at the responder, no mechanism to increase computational expenditure at the initiator is provided. The absence of a proof of work from this recently proposed protocol is conspicuous and exposes the responder to a computational denial of service in the presence of an initiator willing to reveal their IP address. The initiator could engage the responder with a legitimate message 1, then fabricate a bogus message 3 at a minimal computational cost. The responder would have to perform a modular exponentiation before being able to determine that the received message was bogus. The addition of a proof of work, in the form of a client puzzle, would allow the responder to increase the computational

	I		R
	$i \in_R [1, 2, \ldots, q-2]$		$r \in_R [1, 2, \ldots, q-2]$ $sig_{R1} = S_R[g^r, grpinfo_R]$
1)	$N_I' = H(N_I)$	$\xrightarrow{N_I', g^i, ID_R'}$	$Cookie = H(g^r, N_R, N_I', IP_I)$
2)	verify sig_{R1} $K_e = H_{g^{ir}}(N_I', N_R, \text{'1'})$ $K_a = H_{g^{ir}}(N_I', N_R, \text{'2'})$ $K_s = H_{g^{ir}}(N_I', N_R, \text{'0'})$ $sig_I = S_I[N_I', N_R, g^i, g^r, ID_R, sa_I]$ $E1 = \{ID_I, sa_I, sig_I\}_{K_a}^{K_e}$	$\xleftarrow{N_I', N_R, g^r, grpinfo_R, ID_R, Cookie, sig_{R1}}$	
3)		$\xrightarrow{N_I, N_R, g^i, g^r, Cookie, E1}$	$N_I' = H(N_I)$ $Cookie \overset{?}{=} H(g^r, N_R, N_I', IP_I)$ $K_e = H_{g^{ir}}(N_I', N_R, \text{'1'})$ $K_a = H_{g^{ir}}(N_I', N_R, \text{'2'})$ $K_s = H_{g^{ir}}(N_I', N_R, \text{'0'})$ verify and decrypt $E1$ verify sig_I $sig_{R2} = S_R[N_I', N_R, g^i, g^r, ID_I, sa_I, sa_R]$ $E2 = \{sig_{R2}, sa_R\}_{K_a}^{K_e}$
4)	verify and decrypt $E2$ verify sig_{R2}	$\xleftarrow{E2}$	

Figure 3: JFKi Protocol [24]

expenditure of an initiator attempting to mount such an attack - with verification of the puzzle solution only costing a single hash operation.

4.2.3 Gradual authentication

In message 3, the initiator releases the preimage (N_I) to the nonce (N_I') provided in message 1. This binds message 1 and 3 to the same initiator. The initiator must also derive the Diffie-Hellman key (g^{ir}) and the encryption and authentication keys (K_e, K_a) used to protect the contents of message 3.

On receipt of message 3, the responder conducts a range of checks that gradually authenticate the message, prior to conducting an expensive signature verification to strongly authenticate the initiator. First, the responder validates the nonce N_I. Then the cookie is verified to weakly assure the responder that the initiator is reachable at the claimed address. Once these checks complete successfully, the responder then decrypts and verifies the contents of message 3. Finally, the initiator's signature is verified.

The encryption and authentication (via a message authentication code) of message 3 provides the responder assurance that the initiator is willing to commit computational resources to having the protocol proceed. Unlike a proof of work however, the decryption and MAC verification require the responder to incur an equivalent computational cost to the initiator (that of a modular exponentiation) so there is no counterbalancing of computational effort. This technique does provide gradual authentication however, as a failure to correctly decrypt or verify the MAC of a received message 3 allows the responder to detect a possible attack before committing the resources for an expensive signature verification.

4.3 Client-Aided RSA SSL / TLS

Castelluccia et al. [21] observed that the computational expenditure of a responder in the SSL / TLS protocol could be reduced through the adoption of client aided computation. In the SSL / TLS protocol a responder receives a public key encrypted copy of an initiator selected session secret and has to perform an expensive RSA decryption operation in order to generate session keys.

Fortunately, work had already been done on implementing server aided signature generation for resource constrained smart cards [26, 27] and Castelluccia et al. were able to apply the same techniques to have the initiator aid the responder in decrypting the initiator selected session secret. They termed this approach client-aided RSA (CA-RSA). The details of the CA-RSA protocol, which is designed to be compatible with existing SSL/TLS deployments, are presented in Figure 4.

In the SSL three way handshake protocol adopting CA-RSA, the first *client hello* message remains unchanged. The *server hello* message includes the server's certificate and the vector $D = (d_1,d_2,\ldots,d_k)$. The initiator aids responder computation by performing calculations with these values. The client randomly chooses the secret value x, which is used to compute the SSL session key. The secret value x is encrypted with the server's public key component e: $y = x^e \pmod N$. Next, the client uses D to create a new vector Z by computing $z_i = y^{d_i} \pmod N$, for $1 \leq i \leq k$. The client then returns the vector Z and the encrypted session key seed y to the responder in the *client key exchange* message. Once the server receives this message it uses the elements of vector Z to recover x, by computing the values $M_p = \prod_{i=1}^{k} z_i^{f_i} \pmod P$ and $M_q = \prod_{i=1}^{k} z_i^{g_i} \pmod Q$. Finally, the responder recovers x by calculating $M_p n_p + M_q n_q \pmod N$, which is much less computationally expensive than performing the modular exponentiation $y^d \pmod N$. The responder can now derive the session key K_s.

4.3.1 Counterbalancing memory expenditure

As with the unmodified version of TLS, CA-RSA does not make use of cookies to counterbalance memory expenditure. The puzzle construction adopted by the protocol is inadequate to replace the function of a cookie for allowing the protocol responder to remain stateless or serve as a reachability test. Two consequences of this are that the responder must store state on each connection request, making it vulnerable to a memory-based denial of service attack, and secondly, the responder has no way of assessing whether the initiator is using a spoofed IP address.

4.3.2 Counterbalancing computational expenditure

Recognising that while CA-RSA eases responder computational burden it cannot be used as a proof of work, the protocol is supplemented by the addition of a Juels and Brainard [12] style client puzzle to ensure that the responder only attempts to decrypt values from initiators who can provide a proof of work. Unfortunately the puzzle construction specified is in violation of the guidelines specified by Juels and Brainard and is constructed by hashing a random value. A puzzle construction that does not include time, or rely on a time variant responder secret or any connection specific parameters, introduces numerous problems. Firstly, the responder will be unable to know whether the puzzle solution it is verifying is to a puzzle that it issued[3] or if it is a puzzle that has been solved previously unless it stores state. As mentioned earlier, storing state leaves the responder vulnerable to memory-based denial of service attacks. Secondly, the failure to make puzzles time variant provides no mechanism for a responder to defend itself against an initiator that hoards puzzles, generating solutions at its convenience, and then flooding the responder with legitimate puzzle solutions. Finally, the failure to encode connection specific parameters into the puzzle, or to make use of cookies, prevents the responder remaining stateless after message 1 and results in the responder having no assurance that the initiator is reachable at the IP address claimed. Storing state without confirmation of initiator reachability exposes the responder to anonymous memory-based flooding denial of service attacks.

4.3.3 Gradual authentication

Assuming that the responder maintains the significant amount of state required to keep track of issued puzzles, the receipt of a valid puzzle solution could provide some assurance that the initiator has committed computational resources to having the protocol proceed.

While this protocol attempts to counterbalance computational expenditure, the combination of a poorly constructed proof of work, a failure to counterbalance memory expenditure leaves the protocol vulnerable to denial of service attacks. To counterbalance memory expenditure, the responder should adopt the use of cookies, or use a puzzle construction that is consistent with meeting the functional requirements of a cookie.

5 CONCLUSIONS AND FUTURE WORK

Key establishment protocols are particularly vulnerable to denial of service attacks owing to the significant resources they must expend in authenticating initiators and generating the cryptographic keys used for securing ongoing communications. The goal of denial of service resistance in key establishment protocols is to ensure that attackers cannot prevent a legitimate initiator and responder deriving cryptographic keys without expending resources beyond a responder determined threshold.

[3]If an initiator can choose its own puzzles to solve independently of the responder, there would be a great risk of the initiator precomputing a large number of puzzle solutions to use in a denial of service attack.

	I		R
			e, d, N: RSA parameters $f_i, g_i \in_R \{0,1\}^c$ $D = (d_1, d_2, \ldots, d_k)$ s.t. $d \equiv \sum_{i=1}^{k} f_i d_i \bmod P-1$ and $d \equiv \sum_{i=1}^{k} g_i d_i \bmod Q-1$ $n_p = Q(Q^{-1} \bmod P)$ $n_q = P(P^{-1} \bmod Q)$
1)		$\xrightarrow{ID_I, N_I, sa_I}$	$s \in_R \{0,1\}^a$ $t = H(s)$ $puzzle = (t, s_{(b)})$; $s_{(b)}$ = last b bits of s
2)	J s.t. $H(J\|\|s_{(b)}) = t$ $x \in_R \{0,1\}^{48}$ $K_s = H(x, N_I, N_R)$ $y = x^e \bmod N$ $z_i = y^{d_i} \bmod N$ $Z = (z_1, z_2, \ldots, z_k)$	$\xleftarrow{ID_I, ID_R, N_I, N_R, sa_R, puzzle, (e,N), D}$	
3)		$\xrightarrow{ID_I, ID_R, N_I, N_R, sa_R, J, y, Z}$	$H(J\|\|s_{(b)}) \overset{?}{=} H(s)$ $M_p = \prod_{i=1}^{k} z_i^{f_i} \bmod P$ $M_q = \prod_{i=1}^{k} z_i^{g_i} \bmod Q$ $x = y^d =$ $M_p n_p + M_q n_q \bmod N$ $K_s = H(x, N_I, N_R)$
4)		$\xleftarrow{ServerFinish}$	

Figure 4: CA-RSA Protocol [21]

In this paper we have explored the strategies and techniques that permit responders to counterbalance memory expenditure, counterbalance computational expenditure, and to gradually authenticate initiators, thereby determining the level of resources an attacker must commit to disrupting the key establishment protocol and improving the responders resistance to denial of service attacks. The adoption of denial of service resistance techniques in three key establishment protocols was critically analysed with misapplication of techniques identified and recommendations for more effectively applying the techniques made.

Cookies were identified as a technique that can counterbalance memory expenditure and initiate gradual authentication. Correctly constructed cookies allow the protocol responder to remain stateless and serve as a reachability test, providing the responder with assurance that an initiator is able to send and receive messages from a claimed address. Cookie generation must not lead to any state creation, as this will expose the responder to a "cookie crumb" attack.

Proofs of work are hard but tractable problems that can be used by an initiator to prove to a responder that a verifiable level of computational effort has been expended. Proofs of work can be used to counterbalance computational expenditure at the responder and authenticate the commitment of initiator to expending resources to having the protocol proceed. While proofs of work can be constructed from a range of underlying problem, proofs of work based on hash-based constructions are the most prevalent as they are simple to construct and can be verified cheaply. The requirements for overloading cookie functionality into a hash-based client puzzle were presented with the observation that puzzles based on the Aura et al. [13] construction cannot implement the required func-

Table 2: DoS Resistance Techniques in Protocols

Key Exchange Protocol	Mechanisms	Strategies
Modified IKE	Cookie	Counterbalancing Memory
	PoW	Counterbalancing CPU
	Cookie → HASH_I^* → Signature	Gradual Auth.
JFK	Cookie	Counterbalancing Memory
	Nonce → Cookie → MAC → Signature	Gradual Auth.
CA-RSA	Client-Aided Comp.	Counterbalancing CPU
	PoW	Counterbalancing CPU
	PoW → Decrypt secret seed	Gradual Auth.
Photuris [7]	Cookie	Counterbalancing Memory
	Cookie → $E_K\{message\}$ → Signature	Gradual Auth.
HIP [28]	Cookie	Counterbalancing Memory
	PoW	Counterbalancing CPU
	Cookie → PoW → Signature	Gradual Auth.
IKEv2 [29]	Cookie	Counterbalancing Memory
	Cookie → MAC → Signature	Gradual Auth.
Lee & Fung [30]	PoW	Counterbalancing Memory
	PoW	Counterbalancing CPU
	PoW → Signature	Gradual Auth.

tionality, so must be supplemented by a cookie.

We note that the number of key establishment protocols implementing denial of service resistance techniques is limited, with our review of the literature revealing only seven protocols (Table 2 presents a summary). Of those protocols implementing denial of service resistance techniques only two, Host Identity Protocol (HIP) and Modified IKE, use techniques supporting all three strategies: counterbalancing computational expenditure; counterbalancing memory expenditure; and gradual authentication. The protocols implementing all three strategies do not appear to be significantly more complex than the protocols implementing only a subset of the strategies.

The notion of gradual authentication was introduced as a strategy for allowing responders to gain assurance that an attack is not underway and that an initiator is willing to commit computational and memory resources to having the protocol proceed. Specific techniques for gradually authenticating initiators were presented and discussed. A common characteristic of each of the techniques is that they all afford the responder the ability to cheaply verify some aspect of a received message, while fabrication of a message that can pass the responders check is expensive for an attacker. The adversarial models under which each technique is secure, however, are yet to be rigorously analysed. Formalising the adversarial models under which techniques used for gradual authentication may be considered secure is an area for future work.

To exhibit strong denial of service resistance characteristics, we recommend that protocols must adopt techniques to implement all available strategies, including: counterbalancing computational expenditure; counterbalancing memory expenditure; and gradually authenticating requests. Addressing only a subset of these, counterbalancing computational expenditure while storing state prematurely for example, will result in protocols that potentially have residual denial of service vulnerabilities.

Responders should always gain assurance that an initiator is reachable at a claimed address, either via cookies or correctly constructed puzzles. Failure to test reachability leaves the responder vulnerable to attacks from spoofed source addresses.

Finally, there would appear to be a pressing need for developing techniques for quantifying the denial of service resistance a protocol exhibits and how this level of resistance changes with the addition or substitution of techniques. Meadows's cost-based framework [6] for analysing denial of service resistance appears to be a promising approach. Adoption of the framework and analysis of its suitability for informing protocol design choices with respect to the selection of denial of service resistance techniques is also an area of future work.

REFERENCES

[1] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," Standards Track RFC 2401, IETF, Nov 1998, http://www.ietf.org/rfc/rfc2401.txt.

[2] C. E. R. T. (CERT), "Denial-of-Service Attack via ping," 1996, [Online]. Available: http://www.cert.org/advisories/CA-1996-26.html [Accessed: August 2004].

[3] ——, "SYN Flooding Attack," 1996, [Online]. Available: http://www.cert.org/advisories/CA-1996-21.html [Accessed: August 2004].

[4] R. M. Needham, "Denial of Service," in *the 1st ACM conference on Computer and communications security*, Fairfax, Virginia, USA, Dec 1993, pp. 151–153.

[5] J. Leiwo, P. Nikander, and T. Aura, "Towards network denial of service resistant protocols," in *the 15th Annual Working Conference on Information Security (SEC2000)*, vol. 175, Beijing, China, Aug 2000.

[6] C. Meadows, "A Cost-Based Framework for Analysis of DoS in Networks," *Journal of Computer Security*, vol. 9(1/2), pp. 143–164, Jan 2001.

[7] P. Karn and W. A. Simpson, "Photuris: Session-Key Management Protocol," Experimental RFC 2522, IETF, Mar 1999, http://www.ietf.org/rfc/rfc2522.txt.

[8] J. Lemon, "Resisting SYN flood DoS attacks with a SYN cache," in *the BSDCon 2002*, Berkley, CA, USA, 11-14 Feb 2002, pp. 89–97.

[9] T. Aura and P. Nikander, "Stateless Connections," in *International Conference on Information and Communications Security*. Beijing, China: Springer-Verlag, Nov 1997, pp. 87–97.

[10] W. A. Simpson, "IKE/ISAKMP Considered Harmful," *USENIX ;login*, vol. 24, no. 6, dec 1999.

[11] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," in *the 12th Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, 1992, pp. 139 – 147, lecture Notes In Computer Science; Vol. 740.

[12] A. Juels and J. Brainard, "Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks," in *the 1999 Network and Distributed System Security Symposium (NDSS '99)*. San Diego, California, USA: Internet Society Press, Reston, Feb 1999, pp. 151–165.

[13] T. Aura, P. Nikander, and J. Leiwo, "DoS-resistant authentication with client puzzles," in *Security Protocols Workshop 2000*. Cambridge, Apr 2000, pp. 170–181.

[14] X. Wang and M. K. Reiter, "Defending Against Denial-of-Service Attacks with Puzzle Auctions (Extended Abstract)," in *the 2003 IEEE Symposium on Security and Privacy (SP'03)*, Berkeley, CA, USA, 11-14 May 2003, pp. 78–92.

[15] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS 99)*, Sep 1999, also available as http://citeseer.nj.nec.com/238810.html.

[16] K. Matsuura and H. Imai, "Modification of Internet Key Exchange Resistant against Denial-of-Service," in *Pre-Proceeding of Internet Workshop 2000 (IWS2000)*, Feb 2000, pp. 167–174.

[17] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock Puzzles and Timed-release Crypto," Massachusetts Institute of Technology, Cambridge, MA, USA, Technical Report TR-684, 10 Mar 1996.

[18] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten, "New Client Puzzle Outsourcing Techniques for DoS Resistance," in *the 11th ACM Conference on Computer and Communications Security (CCS 2004)*. Washington DC, USA: ACM Press, 2004, pp. 246–256.

[19] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately Hard, Memory-bound Functions," in *the 10th Annual Network and Distributed System Security Symposium*, San Diego, California, USA, 6–7 Feb 2003.

[20] C. Dwork, A. Goldberg, and M. Naor, "On Memory-Bound Functions for Fighting Spam," in *the 23rd Annual International Cryptology Conference (CRYPTO 2003)*, Aug 2003, pp. 426–444.

[21] C. Castelluccia, E. Mykletun, and G. Tsudik, "Improving Secure Server Performance by Re-balancing SSL/TLS Handshakes," Cryptology ePrint Archive, Report 2005/037, 2005, http://eprint.iacr.org/.

[22] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Massachusetts Institute of Technology, Technical Report MIT/LCS/TR-212, January 1979. [Online]. Available: ftp://ftp-pubs.lcs.mit.edu/pub/lcs-pubs/tr.outbox/MIT-LCS-TR-212.ps.gz

[23] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[24] W. Aiello, S. M. Bellovin, M. Blaze, J. Ioannidis, O. Reingold, R. Canetti, and A. D. Keromytis, "Efficient, DoS-resistant, secure key exchange for Internet protocols," in *the 9th ACM conference on Computer and communications security*. Washington, DC, USA: ACM Press, 2002, pp. 48–58.

[25] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," Standards Track RFC 2409, IETF, Nov 1998, http://www.ietf.org/rfc/rfc2409.txt.

[26] T. Matsumoto, K. Kato, and H. Imai, "Speeding Up Secret Computations with Insecure Auxiliary Devices," in *Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, 1988, pp. 497 – 506.

[27] G. Horng, "A Secure Server-Aided RSA Signature Computation Protocol for Smart Cards," *Journal of Information Science and Engineering*, vol. 16, pp. 847–855, 2000.

[28] R. Moskowitz, "The Host Identity Protocol (HIP)," Internet Draft, Internet Engineering Task Force, February 2005, http://www.ietf.org/internet-drafts/draft-ietf-hip-base-02.txt.

[29] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," Internet Draft, IETF, Sep 2004, http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-17.txt.

[30] M. C. Lee and C. K. Fung, "A Public-Key Based Authentication and Key Establishment Protocol Coupled with a Client Puzzle," *Journal of the American Society for Information Science and Technology*, vol. 54(9), pp. 810–823, Jun 2003.