Towards Denial-of-Service-Resilient Key Agreement Protocols[*]

Douglas Stebila[1] and Berkant Ustaoglu[2]

[1] Information Security Institute, Queensland University of Technology, Brisbane, Australia
Email: douglas@stebila.ca

[2] NTT Information Sharing Platform Laboratories, Tokyo, Japan
Email: bustaoglu@cryptolounge.net

April 20, 2009

Abstract

Denial of service resilience is an important practical consideration for key agreement protocols in any hostile environment such as the Internet. There are well-known models that consider the security of key agreement protocols, but denial of service resilience is not considered as part of these models. Many protocols have been argued to be denial-of-service-resilient, only to be subsequently broken or shown ineffective.

In this work we propose a formal definition of denial of service resilience, a model for secure authenticated key agreement, and show how security and denial of service resilience can be considered in a common framework, with a particular focus on client puzzles. The model accommodates a variety of techniques for achieving denial of service resilience, and we describe one such technique by exhibiting a denial-of-service-resilient secure authenticated key agreement protocol. Our approach addresses the correct integration of denial of service countermeasures with the key agreement protocol to prevent hijacking attacks that would otherwise render the countermeasures irrelevant.

1 Motivation

Reliable, fast, and secure communication is essential for commercial success on today's Internet. Slow web pages could motivate clients to switch to an alternative business, leading to a loss in customer base for the service provider. However, maintaining sufficiently powerful servers can be an expensive venture. Servers have limited resources in terms of the amount

[*]Douglas Stebila and Berkant Ustaoglu. Towards Denial-of-Service-Resilient Key Agreement Protocols. In Colin Boyd and Juan Gonzalez-Nieto, editors, *Proc. 14th Australasian Conf. on Information Security and Privacy (ACISP) 2009*, LNCS. Springer, 2009, To appear. EPRINT http://www.douglas.stebila.ca/research/papers/SU09.

of traffic that can be handled, the time required to establish a connection, and the number of active concurrent connections. This effectively bounds the number of connections a company's server can honour in a given period in time.

Malicious parties have recognized that they can benefit by depleting the limited resources of others' servers. *Denial of service attacks* aim to disrupt, destroy, or render services unavailable. A typical denial of service attack exhausts the target's resources. The server is rendered unavailable for honest clients, who then proceed to request similar services from competitors. To prevent malicious requests, a server needs to filter out bogus connection requests and honour those from legitimate clients. Recognizing legitimate clients is difficult. We will view a client as having *legitimate intentions* if it is willing to perform an expensive computation; it still could be malicious, but we may have no further way of distinguishing legitimate from malicious connections with a priori authentication.

Security is an important aspect of online services. Many connections between a client and a server need to be secured against third parties; financial transactions are the most common case. Key agreement is used to produce a shared secret that can be used to encrypt subsequent communication. Key exchange involves computationally expensive algorithms, and hence may dominate server-side run time, limiting the number of clients serviced. This makes key agreement an enticing target for denial of service attacks since a malicious party can easily issue many key agreement requests. Hence, it is advantageous to try to reject as many bogus connections as possible during key agreement. In this paper we are concerned with deterring malicious parties from initiating denial of service attacks based on key agreement, without compromising on security.

Practitioners and standardization bodies have recognized the importance of denial of service resilience, but researchers have been slow to respond with a formal treatment of the subject. In some sense, addressing denial of service resembles the initial approach to key agreement: rather than constructing an overall model, a list of ad hoc goals is selected and then it is shown that a protocol meets those goals. As a result, it is difficult to evaluate the strength and usefulness of denial of service countermeasures when integrated into protocols.

Our Contributions. We give a formal definition of denial of service resilience for key agreement protocols in the context of the extended Canetti-Krawczyk (eCK) model for secure key agreement. Our definition for denial of service resilience is sufficiently strong that it prevents known attacks that arose against protocols once thought to be denial of service resilient. Puzzles have been previously used as a denial of service countermeasure, but in an ad hoc manner. Compared to previous work, our contribution models the careful integration of two orthogonal issues: key agreement security and denial of service resilience.

It is well established that improper use of cryptographic algorithms can render them useless. For example, even the most secure password-based system is of no use if weak passwords are used. Similar reasoning applies for DoS countermeasures: even good solutions are useless if they are not used properly. For example, if a proof-of-work in the form of a puzzle solution does not indicate who is the intended recipient, when the solution was created, or who created the solution, then the door is open for misuse. Without careful integration into the overall protocol, DoS countermeasures may not achieve their goal.

Additionally, we present the DoS-CMQV protocol which is a secure key agreement protocol and uses client puzzles to offer denial of service resilience in our model, showing how to achieve security and denial of service resilience together.

2 Previous Work

Key Agreement. Key agreement is an important cryptographic primitive used for building confidential channels. Designing and analyzing key agreement protocols is a non-trivial task. Formal models, which allow complexity-theoretic security arguments for authenticated key agreement, were first proposed by Bellare and Rogaway [BR04] and Blake-Wilson, Johnson, and Menezes [BWJM97]. The work of Canetti and Krawczyk [CK01a] is one of the most influential extensions to the original models. Their work was later augmented by Krawzcyk [Kra05a] and LaMacchia, Lauter and Mityagin [LLM07] to capture a wider range of desirable security properties; we refer to this as the *extended Canetti-Krawczyk* (eCK) model.

In all of the above models the adversary controls *all* communication links. It is not immediately clear how denial of service can be considered alongside key establishment when the adversary may not deliver messages to destinations. However, even in this setting there are meaningful DoS-related goals that can be incorporated into the model; we discuss our extension in Sect. 3.1.

Denial of Service. There are two main types of denial of service attacks (see [BM03, §1.6.6], for example): *resource depletion* attacks and *connection depletion* attacks. In resource depletion attacks a malicious party attempts to drain a server's computational or memory resources. By contrast, connection depletion attacks aim to exhaust the number of allowed connections to the server. A DoS countermeasure can aim to defend against either or both of these types of attacks.

Distributed denial of service (DDoS) attacks, in which many distributed client computers attack a single server, are of significant concern on the Internet today. These types of attacks are very difficult to defend against. One known technique, which we use in this paper, is to allow a server to adjust its denial of service countermeasure based on the load it experiences. Puzzle auctions [WR03] are one such implementation of tunable puzzles.

Aura and Nikander [AN97] introduced the notion of *stateless connections*, in which stateful connections are transformed into stateless ones by attaching the state information to the message and using a message authentication code for integrity. This gives some protection against denial of service by saving the server from having to store session information until later in the exchange when more assurance is possible.

Meadows [Mea99] offered the first formal framework for denial-of-service-resilient protocols, based on the causal sequencing language of fail-stop protocols of Gong and Syverson [GS95]. To avoid connection depletion, Meadows suggests that each message be authenticated with increasingly complex levels of authentication. Meadows then applies this framework to the Station-to-Station protocol [DvOW92] to identify potential DoS attacks but does not provide a denial-of-service-resilient protocol. An application of Meadow's cost-based framework to the JFK protocol revealed a potential DoS attack, and a solution to this problem was proposed using client puzzles [SGNB06]. This underscores the ability of formal models to reveal flaws and the need for the formalization of denial of service resilience.

Cookies. One of the first techniques used to defend protocols against denial of service attacks was cookies. Introduced in the Photuris protocol (published in 1999 as [KS99] but introduced earlier), *cookies* are small authentication tokens returned by a server upon initial connection by the client. In order for the client to be allowed to continue with the connection, the client must echo the cookie back to the server. The server does not store the cookie, instead using the stateless connection technique to check the authenticity of the cookie which the client includes in subsequent messages. Cookies can be applied in Meadows' framework as an early level of authentication.

Krawczyk's SIGMA protocol [Kra03a] was proposed as a successor of the Internet Key Exchange (IKE) protocol used in IPsec, and was adapted to have denial of service resilience in the form of cookies in its implementation in IKEv2 [Kau05]. Cookies are also used in the Just Fast Keying protocol (JFK) proposed by Aiello et al. [ABB+04]. JFK allows the server to reuse its ephemeral private-public key pair across multiple sessions to reduce the server's computational load, at the expense of increasing the potential damage should an ephemeral private key be leaked.

Cookies are a valuable first-order denial of service countermeasure and have been used extensively as described above. However, they are a weak form of denial of service resilience because they do not require an attacker to do anything other than faithfully relay a previously received cookie.

Protocols using cookies can also be susceptible to other types of attacks. Mao and Paterson [MP02, §2.2] described a denial of service attack against IKEv2. In their attack, a malicious party who controls a popular server $\hat{M}$ can redirect legitimate traffic from $\hat{M}$ towards another target server $\hat{M}'$, thereby effecting a denial of service attack against $\hat{M}'$. The attack only costs $\hat{M}$ bandwidth, not computation or memory, and is resilient to cookie-based DoS countermeasures. This attack is possible because there is no strong binding between the DoS countermeasure and the identity of the server to which the client wishes to connect: we codify this notion in our security criterion DoS-2 in Sect. 3. Despite key agreement and denial of service being orthogonal issues, combining them is no trivial task, as demonstrated by this attack.

Puzzles. Dwork and Naor [DN92] introduced the notion of *client puzzles* to defend against denial of service attacks. A server under a denial of service attack can require clients to find the solution to a puzzle before the server allocates resources: the puzzle should be hard to solve but the solution should be easy to verify. Back [Bac97] and Juels and Brainard [JB99] suggested using a hash function so that a client must perform a large number of operations to find the solution; this is a *computation-bound* puzzle. We build on their approach by specifying how puzzles should be integrated with key agreement. Puzzles where computing the solution is more dependent on memory access time, called *memory-bound puzzles*, have also been suggested for use [ABMW03]; these offer less varied running times across different hardware platforms because memory access times vary less than processor speed. Waters et al. [WJHF04] described how puzzles can be distributed across multiple servers for coordinated access.

Aura, Nikander, and Leiwo [ANL00] gave a framework for using hash function preimages as a denial of service in authentication protocols, and lay out the basic principle that "the client should always commit its resources to the authentication protocol first and the server should be able to verify the client commitment before allocating its own resources". We use this principle to develop a model for denial-of-service-resilient key agreement. The technique of [ANL00] is not sufficient to defend against the attack of Mao and Paterson [MP02]; our approach is.

This principle of clients committing resources before the server does applies well to preemptive DoS countermeasures. The server obtains assurance that the client committed resources, but this is no guarantee that the client will *complete* the request. If a client does not finish a request then what should the actions of the server be? What should a server do if a client takes too long to respond after presenting its proof-of-work? Even though such open connections have important practical significance, preemptive measures do not completely cover the problem of open connections, such as the half-open connections of TCP SYN flood attacks [Edd07]; a common countermeasure is to discard old uncompleted open connections.

3 Modelling Denial of Service Resilience and Security

We begin with an informal description of the goals of a denial-of-service-resilient protocol and then proceed to outline a formal model for integrating denial of service resilience and secure key agreement protocols. While the goals of denial of service resilience and secure key agreement are, as others such as Krawczyk [Kra03a, §2.3] have noted, orthogonal issues, it is useful to be able to discuss them in a common framework. We must be careful to integrate the two issues sufficiently well to avoid the types of attacks proposed by Mao and Paterson [MP02].

Denial of Service Resilience Intuition. We are concerned about the situation in which a malicious party on the network can cause a server to perform many expensive operations (and key agreement is one such expensive operation) for no good reason, eventually consuming all of the server's available resources. But since the server is willing to place itself on the network for the use of all users, how can the server know if it is doing work for a good reason or not? Distinguishing legitimate requests from malicious requests is an essential element of denial of service resilience.

While one can never be certain about the good intentions of another party on the network, it is plausible to believe that a client is making a legitimate request if the client is willing to commit some expensive resources – computation, memory, etc. – to the connection request. However, if a client does do something expensive to prove its good faith, then a good protocol should protect the client from being exploited by a malicious party aiming to steal the client's work. Last but not least, a server should be able to adapt its procedures to resist being flooded by many "honest" requests that may be coordinated for maximum impact.

These ideas lead us to the following five informal criteria for a denial-of-service-resilient protocol:

DoS-1. An uncompromised honest server $\hat{B}$ does not perform any expensive operations with a client unless it is convinced the client is trying to make a legitimate connection.

DoS-2. Moreover, a server $\hat{B}$ does not perform any expensive operations unless it is convinced that the client wants to talk to $\hat{B}$ and not another server $\hat{M}$.

DoS-3. A client $\hat{A}$ who commits significant resources to prove its legitimate intentions cannot have her work stolen: the work that $\hat{A}$ does to convince $\hat{B}$ that it wants to communicate legitimately with $\hat{B}$ cannot convince anyone of anything else.

DoS-4. A malicious party[1] must use a very large amount of resources if it wishes to prepare sufficiently many connection requests and "flood" a server with many valid connection requests.

DoS-5. A server can adjust the amount of work a client has to do in times of higher or lower load.

In Sect. 3.1.2, we give a formal definition of denial of service resilience and describe in Sect. 3.2 how it achieves each of the goals DoS-1 through DoS-4; goal DoS-5 is a property of a particular countermeasure and not of the formal model.

The first two goals aim to protect the server from performing unnecessary expensive operations. While what qualifies as an expensive operation can vary depending on the setting, we identify three main classes of expensive operations for the purposes of denial of service: *memory denial of service* attacks, in which the server is forced to perform slow, expensive memory reads or writes or use large amounts of memory; *computational denial of service* attacks, in which the server is forced to perform operations requiring significant computational time (such as exponentiation, elliptic curve point multiplication, or many simpler operations such as hash function or MAC evaluations); and *transmission denial of service* attacks, in which the server is forced to expend resources available to send and receive communications. In various situations, different notions of expensive can apply; for example, in mobile environments, transmitting and receiving take a lot of time and power.

To achieve denial of service resilience, we require that a client answer a puzzle that takes significant computational or memory resources to solve, but is easy for a server to prepare and verify. The key idea is to tightly bind the puzzles with the identities of the parties involved to prevent attacks in which work can be stolen or redirected. This allows a server to be more convinced of a client's legitimate intention to engage in a key agreement protocol.

Secure Key Agreement Intuition. As noted in Sect. 2, secure key agreement has been extensively studied. The goal of an adversary in the eCK model is to learn any information about the session key established by a pair of uncompromised participants. If an adversary cannot distinguish such a session key from a randomly selected string with non-negligible probability, then the session key is viewed as secure and suitable for use in bulk encryption. The model is formally described in Sect. 3.1 and the definitions of security are given in Sect. 3.1.1.

[1]A malicious party can be a single entity or a collection of entities working together.

3.1 Formal Model Description

Our model is based on the extended Canetti-Krawczyk (eCK) model proposed in [LLM07]. However, instead of all exchanged messages we use a fingerprint of exchanged messages to identify session.

A protocol takes place among n parties $\mathsf{Parties} = \{\hat{A}, \hat{B}, \ldots\}$, where a *party* is a probabilistic polynomial-time Turing machine. In addition to the certified and validated static key pair, each party also possesses static information that is not certified. If non-empty, the non-certified information may be either private or public. Parties are activated via incoming messages, which are then processed within the party. As a result a party either returns its outgoing response or indicates if the processing resulted in failure or success.

Pre-session and Session Creation. An execution of the protocol is called a *session*. A party may receive an incoming *request* to initiate a session via a message (i) $(\hat{A}, \hat{B})$ or (ii) $(\hat{A}, \hat{B}$, "hello"). In the former case $\hat{A}$ is called the *client* or *initiator*, and reacts by creating a separate *session request* $(\hat{B}, \hat{A}$, "hello") designated for $\hat{B}$. In the latter case, $\hat{A}$ is called the *server* or *responder*, and creates a separate *session request* $(\hat{B}, \hat{A}$, "hello") designated for $\hat{B}$. Server $\hat{A}$ selects a fresh (unique within $\hat{A}$) challenge ch and sends $(\hat{B}, \hat{A}$, "hello"$, \mathsf{ch})$ to $\hat{B}$. Within a party the execution of the subroutines between the session request and either accepting or rejecting the request to initiate a session is called a *pre-session.* The model does not prevent a protocol from accepting multiple distinct responses to the same challenge.

The motivation for the pre-session is to include the denial of service countermeasure in the pre-session and leave expensive operations and resources commitment by a server for the session. A session can only be reached after a successful pre-session: in other words, expensive resources will only be committed by the server once the denial of service countermeasure are passed.

A party $\hat{A}$ can be activated to *create* a session with a message of the form (i) $(\hat{A}, \hat{B}$, "hello"$, \mathsf{ch})$ or (ii) $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$. If the activation is of type (i) then $\hat{A}$, who is the initiator, prepares re that passes all protocol conditions and creates an *active* session. The string re must be unique within $\hat{A}$; the outgoing message is $(\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re})$. If the activation is of type (ii) then $\hat{A}$, who is the responder in this case, verifies that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ satisfies the protocol requirements; if so a new active session is created, otherwise the message is ignored. If a responder $\hat{A}$ creates a new session, then the outgoing message is (Ψ, mesg), where mesg is prepared by $\hat{A}$ in accordance with the protocol and Ψ is the *session string identifier*, a string used to identify sessions within $\hat{A}$ and $\hat{B}$, which is derived from $(\mathsf{ch}, \mathsf{re})$ and possibly other publicly known parameters. The conditions imposed on ch and re allow for the derivation of a string unique within both $\hat{A}$ and $\hat{B}$.

Session State. Upon creating a session, $\hat{A}$ also creates a separate *session state* that contains both private and public session-specific information. The private information is needed to derive a secret session key. The public information is $(\hat{A}, \hat{B}, \Psi, \mathsf{role}, \mathsf{otherinfo})$, where $\hat{B}$ is the purported *session peer*; role is either "initiator" or "responder" and $\mathsf{otherinfo}$ is any other public information required by the protocol. Globally the session is identified via $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$ and Ψ identifies the session within $\hat{A}$. For $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$ we call $\hat{A}$ the owner and together $\hat{A}$ and $\hat{B}$ are the *communicating partners* of sid. Sessions $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$ and $\mathsf{sid}^* = [\hat{C}, \hat{D}, \Psi']$ are *matching* if $\Psi = \Psi'$, $\hat{A} = \hat{D}$, and $\hat{C} = \hat{B}$.

As in the eCK model, $\hat{A}$ can be activated to update a session via a message of the form (Ψ, mesg). Upon receipt of such a message, $\hat{A}$ performs validation procedures similar to the eCK model and updates its state. At any stage a session is in exactly one of the following states: *active*, *completed* or *aborted*.

Adversary. The adversary $\mathcal{M}$ is a probabilistic Turing machine that controls *all* communications and party activation via the query $\mathsf{Send}(\cdot)$. Parties present $\mathcal{M}$ with their outgoing messages. Leakage of private information to $\mathcal{M}$ is modelled via the following adversary queries:

- $\mathsf{StaticKeyReveal}(\hat{A})$: $\mathcal{M}$ obtains $\hat{A}$'s static private key.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$: $\mathcal{M}$ obtains the ephemeral private key of $\hat{A}$ in session $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$.
- $\mathsf{SessionKeyReveal}(\mathsf{sid})$: If sid has completed then $\mathcal{M}$ obtains the session key in sid.
- $\mathsf{Establish}(\hat{M}, M)$: $\mathcal{M}$ registers an *adversary-controlled* party $\hat{M}$ with static public key $M \in \mathcal{G}$. If a party is not adversary-controlled it is said to be *honest*.
- $\mathsf{DoSExpose}(\hat{A})$: $\mathcal{M}$ obtains the non-certified private information belonging to an honest $\hat{A}$, excluding the session-related ephemeral private keys. Parties against which this query was issued are called *DoS-exposed*, otherwise they are called *DoS-unexposed*.

The role of the new $\mathsf{DoSExpose}$ query in our model is to allow us to identify the parties which ought to still be resilient to denial of service attacks, namely those parties which are DoS-unexposed. For example, adversary-controlled parties are not relevant to the DoS portion of the protocol. Moreover, a separate $\mathsf{DoSExpose}$ query allows us to separate denial of service resilience from key agreement security: compromise of key agreement secrets can be an orthogonal issue to compromise of denial of service.

3.1.1 Key Agreement Security Definitions.

Security is defined via indistinguishability. At any time during the experiment $\mathcal{M}$ can make one special query, $\mathsf{Test}(\mathsf{sid})$, to a session sid that must remain fresh throughout the experiment. The goal of $\mathcal{M}$ is to guess whether the response to the query is sid's key or a random key.

Definition 1 (Fresh session) *Let* sid *be the identifier of a completed session, owned by an honest party* $\hat{A}$ *with peer* $\hat{B}$*, who is also honest. Let* sid^* *be the identifier of the matching session of* sid*, if it exists. Then* sid *is* fresh *if none of the following conditions hold:*

1. $\mathcal{M}$ *issued* $\mathsf{SessionKeyReveal}(\mathsf{sid})$ *or* $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$ *(if* sid^* *exists).*
2. sid^* *exists and* $\mathcal{M}$ *issued one of the following: either*
   (a) *both* $\mathsf{StaticKeyReveal}(\hat{A})$ *and* $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$*, or*

    (b) *both* $\mathsf{StaticKeyReveal}(\hat{B})$ *and* $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$.

3. sid^* *does not exist and* $\mathcal{M}$ *issued one of the following: either*

    (a) *both* $\mathsf{StaticKeyReveal}(\hat{A})$ *and* $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$*, or*

    (b) $\mathsf{StaticKeyReveal}(\hat{B})$.

Definition 2 (Secure key agreement protocol) *A key agreement protocol is* secure *if the following conditions hold: (i) two honest parties that complete matching sessions then, except with negligible probability they compute the same session key; and (ii) no polynomially bounded adversary* $\mathcal{M}$ *can distinguish the session key of a fresh session from a randomly chosen session key with probability greater than* $\frac{1}{2}$ *plus a negligible fraction (in the security parameter).*

3.1.2 Denial of Service Definitions.

In the protocol there is a test that the server performs on some of the messages received to determine if the client has done sufficient work to merit the server performing expensive operations. The client's work, modelled by a puzzling relation, is used to define the protocol's denial of service resilience.

Definition 3 (Puzzling relation) *Let* $\mathsf{Challenges}$ *and* $\mathsf{Responses}$ *be sets. A relation* $\mathcal{R} \subseteq \mathsf{Parties} \times \mathsf{Parties} \times \mathsf{Challenges} \times \mathsf{Responses}$ *is a* puzzling relation *if*

1. *deciding if* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ *is "easy", and*

2. *given* $\hat{A}, \hat{B}, \mathsf{ch}$*, and an oracle* U *that, on input* $(\hat{A}', \hat{B}', \mathsf{ch}')$*, returns* re' *with* $(\hat{A}', \hat{B}', \mathsf{ch}', \mathsf{re}') \in \mathcal{R}$*, it is "hard" to produce* re *such that* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ *and* re *was not a response generated by the oracle* U *upon input* $(\hat{A}, \hat{B}, \mathsf{ch})$.

The notion of "expensive operation", "easy", and "hard" will depend on the application context. Although we have left these notion vague, they can be formalized, for example as a proof of work [JJ99]. We omit this formalization as the focus of our work is on the integration of denial of service resilience techniques into key agreement protocols, not the construction of suitable puzzles.

Definition 4 (Acceptable pre-session) *A pre-session* $[\hat{A}, \hat{B}, \mathsf{ch}]$ *is an* acceptable pre-session *for* $\hat{B}$ *if* $\hat{B}$ *generated* ch.

Definition 5 (Denial-of-service-resilient protocol) *Let* $\mathcal{R}$ *be a puzzling relation. A protocol* Π *is* denial-of-service-resilient *if the following hold for every DoS-unexposed server* $\hat{B}$*:*

1. $\hat{B}$ *only performs expensive operations (a) in a session, or (b) for some (low frequency) periodic update of its non-certified private information* ρ*, and*

2. $\hat{B}$ *only establishes a session* $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$ *if the pre-session* $[\hat{A}, \hat{B}, \mathsf{ch}]$ *was an acceptable pre-session for* $\hat{B}$ *and* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$.

Note that we have explicitly avoided merging Definitions 4 and 5 (as Definition 4 could be part of Condition 2 of Definition 5) because we acknowledge that there may be situations that require a different notion of acceptable pre-session, for example one-pass key agreement protocols where both ch and re are generated by the initiator. In particular, it appears to suffice that $\hat{B}$ has an assurance that ch was generated independently at random before re.

3.2 Model Implications

In this section we explain how our formal model of denial of service resilience in Definition 5 satisfies the informal goals for denial of service resilience of Sect. 3.

DoS-1. *An uncompromised honest server $\hat{B}$ does not perform any expensive operations with a client unless it is convinced the client is trying to make a legitimate connection.*

By condition 1 of Definition 5, a server $\hat{B}$ does not perform any expensive operation with a client until it has established a session, and by condition 2 it will not establish a session until it received a response to its challenge that satisfies the puzzling relation. In order to satisfy the puzzling relation, the client must do a significant amount of work because of condition 2 of Definition 3; by doing this work, the client convinces the server that it is trying to make a legitimate connection. Moreover, since sessions are unique within a party, replay attacks of legitimate connection requests are prevented.

DoS-2. *Moreover, a server $\hat{B}$ does not perform any expensive operations unless it is convinced that the client wants to talk to $\hat{B}$ and not another server $\hat{M}$.*

Condition 2 of Definition 3 allows us to meet this criterion: even if an adversary obtains any tuple $(\hat{A}, \hat{M}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$, the tuple $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ is unlikely to be in $\mathcal{R}$ and moreover it remains hard to produce a response re' such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}') \in \mathcal{R}$. Since it is hard to create such a tuple given oracle access to $\mathcal{R}$, then it is still hard to construct any tuple in $\mathcal{R}$ without oracle access.

Our approach avoids the attack of Mao and Paterson [MP02] against IKEv2 in which an attacker can redirect traffic from her server towards other servers and can cause the receiving server to deplete its connection resources at low expense to the attacker. That attack is possible because there is no cryptographic binding between the denial of service countermeasure and the identities of the parties involved. By including the names of the client and server in the puzzling relation, a server $\hat{B}$ can be assured that whoever solved the puzzle intended to communicate with $\hat{B}$.

DoS-3. *A client $\hat{A}$ who commits significant resources to prove its legitimate intentions cannot have her work stolen: the work that $\hat{A}$ does to convince $\hat{B}$ that it wants to communicate legitimately with $\hat{B}$ cannot convince anyone of anything else.*

Suppose $\hat{B}$ is a DoS-unexposed server and suppose an honest client $\hat{A}$ starts a pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$, and then finds a value re such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$. The client wishes that the response value should not be useful to anyone else trying to establish a session; in other words, no one should be able to steal $\hat{A}$'s work and use it in another pre-session.

Suppose $[\hat{A}', \hat{B}', \mathsf{ch}']$ is another pre-session. Given these values, it is hard to produce re' such that $(\hat{A}', \hat{B}', \mathsf{ch}', \mathsf{re}') \in \mathcal{R}$, even with help from another pre-session such as $[\hat{A}, \hat{B}, \mathsf{ch}]$, because $\mathcal{R}$ is a puzzling relation. The help given by the other pre-session can be modelled as one response of the oracle U in Definition 3 for another pre-session, and this is of no help in a puzzling relation. Thus, an honest client's work in solving a puzzle is of no use to anyone else responding to a different challenge, or with a different server, or with a different user name.

If the adversary $\mathcal{M}$ simply relays $\hat{A}$'s entire response and then participates in $\hat{A}$'s place, the server will proceed with key agreement but this session will ultimately fail, since it is secure in the sense of Definition 2 key agreement protocols, and thus $\mathcal{M}$ cannot complete a session masquerading as $\hat{A}$.

DoS-4. *A malicious party must use a very significant amount of resources if it wishes to prepare sufficiently many connection requests and "flood" a server with many valid connection requests.*

As noted in DoS-1, a server will only perform expensive operations if it has been convinced that the client is trying to make a legitimate connection, meaning the client has solved an instance of the puzzling relation, which is "hard" and requires a significant amount of resources. Suppose that for an attacker to start a session requires t steps of computation (e.g., t may be the number of cycles it takes to solve a computationally bound puzzling relation), and a server has enough computational resources to support n connections per second. Then, roughly speaking, an attacker's computers must be able to perform tn steps of computation per second to sustainably render the server unavailable through a denial of service attack, which may require distributed resources. By registering many dishonest parties the above attack can be incorporated in our model. While not completely defending against such powerful distributed attacks, we can at least allow the amount of denial of service resilience to be tuned in the event of heavy traffic. On the other hand what our model assures is that the adversary cannot *use* honest parties to mount such distributed attacks. For example, a DoS-resilient protocol guards against the attack where the adversary registers a single malicious party $\hat{M}$, initiates pre-sessions between honest parties and $\hat{M}$, and then forwards messages from the honest parties to create many sessions at an honest server. That is, the model defends against Mao-Patterson type of attacks.

Consider also the case of *replay attacks*, in which an attacker resends the same message many times to a server. Suppose in particular that an attacker replays a response value re for a pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$. This set of values leads to the server session $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$, which already exists in the server. Since sessions identifiers are unique in the model, the server will not start a new session and hence commit no new resources as a result of this replay. This requires the server to store a table of session identifiers, but this does not result in a denial of service attack in theory since the server commits memory resources for the entries in the table only once the puzzling relation has been passed. To limit the size of the table, the server could change the non-certified information ρ periodically. When receiving a previous challenge and response, an entirely acceptable action would be for the server to respond to the replay with the same response it gave previously; this prevents puzzle stealing attacks where the adversary responds to a puzzle faster than a legitimate client. Now, if the attacker were to compute a different response value re' for the pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$, then the server would commit new resources to the new session, but this is acceptable since the attacker solved the puzzling relation, just as a legitimate client must.

Since in the Canetti-Krawczyk model we allow the adversary to control the delivery of messages, an adversary may choose not to deliver the final message from the client to the server and leave the server with an incomplete session (similar to the half-open connections of TCP SYN flood attacks [Edd07]). Our model does not view this as a denial of service attack, because the server has been assured that the other party performed many expensive computations to create the connection. This type of attack can be mitigated, without affecting the security assurance nor preventive DoS countermeasures, by conventional server policies to deal with open connections that are not completed for a predefined period of time. Such countermeasures can be separately addressed once the server is assured about the correct connection between the proof of work, the client's identity and the client's intended recipient – exactly what our model provides.

4 A Secure DoS-Resilient Key Agreement Protocol

Our DoS-CMQV protocol, given in Fig. 1, is an adaptation of the CMQV [Ust08] secure authenticated key agreement protocol. We use the problem of finding preimages for a random hash function as the expensive puzzle at the heart of the puzzling relation that a client needs to solve.

The notation $L[i]$ refers to the ith component in the tuple L. H_0 and H_1 are random hash functions [BR04] that return bit strings; all other hash functions return random integers between 1 and q, the order of the group $\mathcal{G}$ generated by g. We use $x_{[1\ldots w]}$ to denote the first w bits of x. We note that in practice H_1 should be chosen so as to be unique to the protocol so that puzzles cannot be outsourced to another protocol; for example, $H_1(\ldots) = \mathsf{SHA\text{-}256}(\text{"DoS-CMQV"}, 1, \ldots)$.

4.1 Security Analysis

Theorem 1 *If $H_0, H_1, \ldots, H_5$ are random oracles [BR04], and $\mathcal{G}$ is a group where the Gap Diffie-Hellman (GDH) assumption [OP01] holds, then DoS-CMQV is a secure key agreement protocol.*

Argument. The DoS-CMQV security argument is similar to the argument presented for CMQV in [Ust08]. We proceed to outline the argument. Verifying condition 1 of Definition 2 is straightforward. It remains to verify condition 2.

In the model here parties possess additional (non-certified) private information ρ, which the adversary can obtain via DoSExpose query. For each of the events in the analysis of CMQV, the solver establishes and simulates the parties similar to the CMQV analysis. The main difference is that when parties are established the solver selects randomly the value ρ for each party. The DoSExpose queries are answered faithfully and they do not affect the freshness of the session. Since the new adversary query is not relevant to the security analysis of the events, the solver can transform the DoS-CMQV adversary to a GDH solver with similar success and running time as a CMQV adversary. Hence a polynomially bounded DoS-CMQV adversary contradicts the assumptions in the theorem. □

DoS-CMQV with security parameter λ			
	Client $\hat{A}$		Server $\hat{B}$
0.	$g, a, A = g^a, B$		$g, b, B = g^b, A, \rho \in_R \{0,1\}^\lambda$
1.		$\xrightarrow{\text{“hello”},\hat{A},\hat{B}}$	$i \in_R \{0,1\}^\lambda$
2.			$j = H_0(\rho, \hat{A}, \hat{B}, i)$
3.	store $\tilde{x} \in_R \{0,1\}^\lambda$	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} = (i, j)$
4.	$x = H_2(\tilde{x}, a), X = g^x$		
5.	find ℓ s.t. $H_1(\hat{A}, \hat{B}, \mathsf{ch}, X, \ell)_{[1\ldots 20]} = 0\ldots 0$		
6.	$\mathsf{re} = (X, \ell), \Psi = (\mathsf{ch}, \mathsf{re})$		
7.	establish session $[\hat{A}, \hat{B}, \Psi]$	$\xrightarrow{\hat{A},\mathsf{ch},\mathsf{re}}$	verify $\mathsf{ch}[2] = H_0(\rho, \hat{A}, \hat{B}, \mathsf{ch}[1])$
8.			verify $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots 20]} = 0\ldots 0$
9.			establish unique session $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$
10.			store $\hat{A}, \Psi = (\mathsf{ch}, \mathsf{re})$
11.			$X = \mathsf{re}[1]$
12.			verify $X \in \mathcal{G}$
13.			$\tilde{y} \in_R \{0,1\}^\lambda, y = H_2(\tilde{y}, b)$
14.			store $Y = g^y$
15.			$d = H_3(X, \hat{A}, \hat{B}), e = H_3(Y, \hat{A}, \hat{B})$
16.			$\sigma = (XA^d)^{y+eb}$
17.			store $M_1 = H_4($“server finished”, $\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
18.			store $M_2 = H_4($“client finished”, $\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
19.	verify $Y \in \mathcal{G}$	$\xleftarrow{\Psi,Y,M_1}$	store $K = H_5(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
20.	$d = H_3(X, \hat{A}, \hat{B}), e = H_3(Y, \hat{A}, \hat{B})$		
21.	$\sigma = (YB^e)^{x+da}$		
22.	verify M_1		
23.	$M_2 = H_5($“client finished”, $\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$		
24.	$K = H_5(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$	$\xrightarrow{\Psi,M_2}$	verify M_2

Figure 1: DoS-CMQV: A denial-of-service-resilient adaptation of the CMQV protocol.

4.2 Denial of Service Resilience Analysis

In this section we show that the DoS-CMQV protocol given in Fig. 1 is denial-of-service-resilient according to Definition 5. Since this definition (and the related definition of a puzzling relation) includes the intentionally vague terms “expensive operation”, “easy”, and “hard”, we need to define what these terms mean for a concrete instantiation of the definition.

For our purposes, an *expensive operation* is one of the following operations: storing a per-connection or per-session value in memory (other than a long-term value), performing a group exponentiation, or making a large number of calls (say, more than 2^{10}) to a hash oracle.

We first establish, via the following lemma, that the relation used in the DoS-CMQV protocol is a puzzling relation and then show that our protocol is resilient to denial of service attacks

Lemma 1 *Let $\mathcal{R}$ be the relation defined such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ if and only if $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots 20]} = 0\ldots 0$, where H_1 is a random hash function. Then $\mathcal{R}$ is a puzzling relation, where "hard" means requiring approximately 2^{20} hash function queries on average, and "easy" is something that is not an "expensive operation" as defined above.*

Argument. Deciding membership in $\mathcal{R}$ is easy for a particular tuple because it involves only a single call to H_1.

Moreover, given $\hat{A}$, $\hat{B}$, and a random ch, producing a value re such that $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots 20]} = 0\ldots 0$ is hard and requires approximately 2^{20} hash oracle queries on average. To find such an re requires finding a preimage for the random hash function. The oracle U helps us find other preimages of H_1. Our task, then is to find a preimage of the correct format involving $\hat{A}$, $\hat{B}$, ch. But since H_1 is a random hash function, other outputs do not help in finding a preimage for this input. Since H_1 is a random hash function outputting 20 bits, this is a hard task that requires approximately 2^{20} queries on average. $\square$

This hash puzzle is similar to the partial inversion proof of work (PIPOW) problem of Jakobsson and Juels [JJ99, §3.1]. By their Claim 1, we know that any prover $\hat{A}$ with memory bounded by m who performs on average at most w steps of computation and is given $(\hat{A}, \hat{B}, \mathsf{ch})$ can find a response re such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ with probability at most $p + o(m/2^{20})$ where $p = 1/(2^{20} - w)$.

Theorem 2 *The DoS-CMQV protocol is a denial-of-service-resilient protocol, where "easy", "hard", and "expensive operation" are defined as above.*

Argument. By the Lemma above, $\mathcal{R}$ is a puzzling relation.

Let $[\hat{A}, \hat{B}, \mathsf{ch}]$ be a pre-session. According to the protocol, $\hat{B}$ does not perform any expensive operation until line 10, which is not reached unless the server's checks on lines 7 and 8 are passed and a new session is established on line 9.

If the check on line 8 is passed, namely if $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}[1], \mathsf{re}[2])_{[1\ldots 20]} = 0\ldots 0$, then $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$. If the check on line 7 is passed, namely if $\mathsf{ch}[2] = H_0(\rho, \hat{A}, \hat{B}, \mathsf{ch}[1])$, then, except with negligible probability, $\mathsf{ch}[2]$ was generated only be someone who knew both ρ and $\mathsf{ch}[1]$. Since $\hat{B}$ is a DoS-unexposed party, no $\mathsf{DoSExpose}(\hat{B})$ query could have been issued and since ρ is only ever used as an input to a random oracle, only $\hat{B}$ knows ρ. Thus, $[\hat{A}, \hat{B}, \mathsf{ch}]$ is an acceptable pre-session.

Hence, $\hat{B}$ establishes a session only if the corresponding pre-session is acceptable and the tuple is in the puzzling relation. Note that since sessions must be unique within a party, the server only performs these expensive operations once per session. Thus, DoS-CMQV is a denial-of-service-resilient protocol. $\square$

Tuning the Puzzling Relation. The puzzle used in DoS-CMQV can be tuned by the server based on its load. The client must find a hash function preimage; for concreteness, we have specified that the first 20 bits should be zeros, but the length could be a parameter w set by the server depending on its current load. The server would need to include w in the computation of j on line 2, return w as part of ch on line 3, and include w in the check on line 7 to avoid spoofing.

In practice, H_1 could be implemented by using a standard cryptographic hash function, such as SHA-1, and truncating the output to the first w bits. In times of light load, the server could require that clients truncate only to the first 5 or 10 bits of output, but in heavier load could require that clients truncate to 20 or 25 bits of output to make the cost of mounting a denial of service attack higher. It takes just under 3 seconds to perform 2^{20} SHA-1 evaluations on one core of our 2 GHz Intel Core 2 Duo processor using OpenSSL $0.9.7\ell$. This may be an acceptable computational burden for the client in many scenarios.

5 Other Denial of Service Constructions

Memory-Bound Puzzling Relations. While the protocol given in Sect. 4 uses a puzzling relation based on finding preimages in a hash function, other types of puzzling relations can be used, demonstrating the flexibility of our framework. Abadi et al. [ABMW03], for example, described puzzles in which memory access time provides an expected lower bound on the time it takes to solve the puzzle, removing disparities in processor speed between large computers and small devices. However, care must be taken in choosing parameters for memory-bound puzzles: the cost incurred by a server in setting up one of these memory-bound puzzles, while much less expensive than the cost incurred by a client solving the puzzle, can still be significant. For the memory-bound puzzles of [ABMW03], it took a 2.4 GHz Pentium 4 server approximately 2^{-7} seconds to create a puzzle that takes approximately 2^2 seconds to solve. By comparison, the time to do one 1024-bit modular exponentiation on a computer of similar speed is less: only 2^{-9} seconds.

JFKi. In the JFKi protocol of [ABB$^+$04, §2.3], the denial of service resilience goal for the server is to avoid expensive operations unless the client performs resource-heavy operations, namely group exponentiations. There are two main ideas used: reuse of ephemeral public keys and use of a keyed hash function. The purpose of reusing ephemeral public keys is to distribute the cost of an expensive operation across multiple sessions. This allows the authors to argue the client must perform her share of the work first, in terms of bearing the cost of establishing a round communication trip. The keyed hash function is used by the server to verify that the client indeed executed the round. Note that the server does not need to dedicate any resources to verify the challenge was created by the server. This can be viewed as the pre-session stage of the protocol since the goal of the first round trip is to filter out bogus connections.

JFKi can be described in our model of denial of service resilience, but with weak definitions of "hard" in the puzzling relation. In the implied puzzling relation in JFKi, the client must echo back to the server all the received values and the preimage of the client's nonce. If the puzzling relation test passes, then the server establishes a new session and computes the shared Diffie-Hellman key.

The problem with JFKi's puzzling relation is that there is no binding between the client's ephemeral public key and the solution to the puzzling relation. A dishonest client can use the same solution to the puzzling relation with different ephemeral public keys (and it can generate these ephemeral public keys very cheaply, for example, by generating g^i, $g \cdot g^i = g^{i+1}, g \cdot g^{i+1} = g^{i+2}, \ldots$) to cause the server to perform many exponentiations with little cost to the client. Thus, given the JFKi puzzling relation and a single solution to the puzzling relation, generating more solutions is easy, contradicting Condition 2 of Definition 3.

Hence, JFKi does not satisfy informal goal DoS-4: the protocol is not resilient to flooding attacks. DoS-CMQV avoids this problem: producing a solution to the puzzling relation means finding a preimage in the hash function and the values cannot be repeated if the server is to establish a new session.

One approach to fixing the denial of service resilience of JFKi was given by Smith et al. [SGNB06]. They note that JFKi is not denial-of-service-resilient when analyzed under Meadows' framework [Mea99]. They use a hash function preimage puzzle as well to bind the puzzle solution to the key exchange session at hand. There construction still preserves a fundamental design characteristic of JFKi: the responder must reuse its ephemeral private key in order to achieve denial of service resilience, preventing full freshness in the Canetti-Krawczyk model.

Host Identity Protocol. The Host Identity Protocol (HIP) [MNJH04] was designed to offer protection against denial of service attacks. HIP (in [MNJH04, §4.1.1]) uses a similar puzzling relation to that of Sect. 4: the client must find a preimage in SHA-1 such that the k lowest-order bits of the output are zero. HIP includes the identities of the initiator and responder in the hash function computation as we have done. Our model provides a theoretical interpretation of the security of HIP against denial of service attacks and the value of including the client and server identities in the hash function computation.

6 Conclusion and Open Problems

We have given the first formal definition of denial of service resilience for secure key agreement protocols. Our model uses puzzles solved by the client as an indication of interest in a legitimate connection, and a variety of puzzles, both memory-bound and computation-bound, can be used. We described a protocol, DoS-CMQV, that provides resilience to denial of service attacks and offers secure key agreement. Additionally, we analyzed the existing JFKi and HIP protocols to compare their notions of denial of service resilience.

Denial of service resistance countermeasures often depend on the freshness of challenges. Our model could be extended to explicitly consider the update of puzzle freshness and how past puzzles affect current denial of service resilience.

This new framework for analyzing denial of service resilience can be applied in conjunction with other goals for key agreement protocols. For example, the JFK protocols [ABB$^+$04] aim to offer privacy features: JFKi protects the initiator's identity and JFKr protects the responder's identity. Future work could involve designing denial-of-service-resilient protocols with similar privacy measures.

This model can also be applied to give denial of service resilience to other types of key agreement protocols, for example password-authenticated key agreement protocols. Adapting this technique for use in IPsec or TLS would provide denial of service resilience in important Internet protocols.

Acknowledgements.

This work was performed while the authors were at the University of Waterloo. D.S. was supported by an NSERC Canada Graduate Scholarship. The authors are grateful for the helpful advice of Alfred Menezes and Ian Goldberg.

References

[ABB+04] William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just Fast Keying: Key agreement in a hostile Internet. *ACM Transactions on Information and System Security*, **7**(2):1–30, May 2004. DOI:10.1145/996943.996946.

[ABMW03] Martín Abadi, Michael Burrows, Mark Manasse, and Ted Wobber. Moderately hard, memory-bound functions. In *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 2003*. Internet Society, 2003. URL `http://www.isoc.org/isoc/conferences/ndss/03/proceedings/`.

[AN97] Tuomas Aura and Pekka Nikander. Stateless connections. In Yongei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Proc. 1st International Conference on Information and Communications Security*, *LNCS*, volume 1334, pp. 87–97. Springer, November 1997. DOI:10.1007/BFb0028465.

[ANL00] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DOS-resistant authentication with client puzzles. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *SECPROT*, *LNCS*, volume 2133, pp. 170–177. Springer, 2000. DOI:10.1007/3-540-44810-1_22. URL `http://research.microsoft.com/en-us/um/people/tuomaura/Publications/aura-nikander-leiwo-protocols00.pdf`.

[Bac97] Adam Back. A partial hash collision based postage scheme, 1997. URL `http://www.hashcash.org/papers/announce.txt`.

[BM03] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[BR04] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In Pfitzmann and Liu [PL04], pp. 62–73. DOI:10.1145/168588.168596.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *Cryptography and Coding – 6th IMA International Conference*, *LNCS*, volume 1355. Springer, 1997. DOI:10.1007/BFb0024447.

[CK01a] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances* *in Cryptology – Proc. EUROCRYPT 2001*, *LNCS*, volume 2045, pp. 453–474. Springer, 2001. DOI:10.1007/3-540-44987-6_28. Full version available as [CK01b].

[CK01b] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels, 2001. EPRINT `http://eprint.iacr.org/2001/040`. Extended abstract published as [CK01a].

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology – Proc. CRYPTO '92*, *LNCS*, volume 740, pp. 139–147. Springer, 1992. DOI:10.1007/3-540-48071-4_10.

[DvOW92] Whitfield Diffie, Paul van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, **2**(2):107–125, June 1992. DOI:10.1007/BF00124891.

[Edd07] Wesley M. Eddy. TCP SYN flooding attacks and common mitigations, August 2007. URL `http://www.ietf.org/rfc/rfc4987.txt`. RFC 4987.

[GS95] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pp. 44–55, September 1995. URL `http://citeseer.ist.psu.edu/article/gong94failstop.html`.

[JB99] Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 1999*, pp. 151–165. Internet Society, 1999. URL `http://www.isoc.org/isoc/conferences/ndss/99/proceedings/`.

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Bart Preneel, editor, *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*, *IFIP Conference Proceedings*, volume 152, pp. 258–272. Kluwer, 1999. URL `http://www.rsa.com/rsalabs/node.asp?id=2049`.

[Kau05] Charlie Kaufman. Internet Key Exchange (IKEv2) protocol, December 2005. URL `http://www.ietf.org/rfc/rfc4306.txt`. RFC 4306.

[Kra03a] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *Advances in Cryptology – Proc. CRYPTO 2003*, *LNCS*, volume 2729, pp. 400–425. Springer, 2003. DOI:10.1007/b11817. Full version available as [Kra03b].

[Kra03b] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols, 2003. URL `http://www.ee.technion.ac.il/~hugo/sigma.ps`. Short version published as [Kra03a].

[Kra05a] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – Proc. CRYPTO 2005*, *LNCS*, volume 3621, pp. 546–566. Springer, 2005. DOI:10.1007/11535218_33. Full version available as [Kra05b].

[Kra05b] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol, 2005. EPRINT `http://eprint.iacr.org/2005/176.pdf`. Extended abstract published as [Kra05a].

[KS99] Phil Karn and William Allen Simpson. Photuris: Session-key management protocol, March 1999. URL `http://www.ietf.org/rfc/rfc2522.txt`. RFC 2522.

[LLM07] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *First International Conference on Provable Security (ProvSec) 2007*, *LNCS*, volume 4784, pp. 1–16. Springer, 2007. DOI:10.1007/978-3-540-75670-5_1. EPRINT `http://eprint.iacr.org/2006/073`.

[Mea99] Catherine Meadows. A formal framework and evaluation method for network denial of service. In *Proc. 1999 IEEE Computer Security Foundations Workshop (CSFW)*, p. 4. IEEE Computer Society Press, 1999. DOI:10.1109/CSFW.1999.779758.

[MNJH04] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas R. Henderson. Host identity protocol. Online, February 2004. URL `http://tools.ietf.org/html/draft-moskowitz-hip-09`. Internet-Draft.

[MP02] Wenbo Mao and Kenneth G. Paterson. On the plausible deniability feature of Internet protocols. Manuscript, 2002. URL `http://citeseer.ist.psu.edu/678290.html`.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Public Key Cryptography (PKC) 2000*, *LNCS*, volume 1992, pp. 104–118. Springer, 2001. DOI:10.1007/3-540-44586-2_8.

[PL04] Birgit Pfitzmann and Peng Liu, editors. *Proc. 11th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2004.

[SGNB06] Jason Smith, Juan Gonzalez-Nieto, and Colin Boyd. Modelling denial of service attacks on JFK with Meadows's cost-based framework. In Rajkumar Buyya, Tianchi Ma, Reihaneh Safavi-Naini, Chris Steketee, and Willy Susilo, editors, *Proc. 4th Australasian Information Security Workshop – Network Security (AISW-NetSec) 2006*, *CRPIT*, volume 54, pp. 125–134. Australian Computer Society, 2006. URL `http://crpit.com/confpapers/CRPITV54Smith.pdf`.

[Ust08] Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, **46**(3):329–342, March 2008. DOI:10.1007/s10623-007-9159-1. EPRINT `http://eprint.iacr.org/2007/123.pdf`.

[WJHF04] Brent Waters, Ari Juels, J. Alex Halderman, and Edward W. Felten. New client puzzle outsourcing techniques for dos resistance. In Pfitzmann and Liu [PL04], pp. 246–256. DOI:10.1145/1030083.1030117.

[WR03] Xiaofeng Wang and M.K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proc. 2003 IEEE Symposium on Security and Privacy (SP'03)*, pp. 78–92. IEEE Press, 2003. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=27002&arnumber=1199329`.