Counter-Flooding: DoS Protection for Public Key Handshakes in LANs

Yves Igor Jerschow Björn Scheuermann Martin Mauve

Institute of Computer Science, Heinrich Heine University, Düsseldorf, Germany
{jerschow, scheuermann, mauve}@cs.uni-duesseldorf.de

***Abstract*—The majority of security protocols employ public key cryptography for authentication at least in the connection setup phase. However, verifying digital signatures is an expensive task compared to symmetric key operations and may become the target for Denial of Service (DoS) attacks, where the adversary floods the victim host with fake signature packets trying to overload it. In this paper we present counter-flooding, a new defense mechanism against DoS attacks which exploit the lack of initial address authenticity in LANs. A benign host having a signature packet addressed to a host which is currently under attack ensures the processing of its packet by flooding copies of this packet for a short period of time itself. The key idea is for the victim host to verify only a fixed number of signatures per time period without becoming overloaded and to select those packets for verification which have the largest number of duplicates. Under weak assumptions we prove that the packet from the benign host will be among them. We derive bounds for our counter-flooding mechanism to succeed and perform experiments with Ethernet switches to study the bandwidth division between concurrent flows under overload conditions.**

I. Introduction

The application of security protocols to protect against eavesdropping, spoofing, and Denial of Service (DoS) attacks is prevalent not only in the Internet, but also in Local Area Networks (LANs). Usually these protocols employ public key cryptography during the setup phase—in particular digital signatures along with certificates—to perform authentication and exchange keying material. Examples of such LAN security protocols are: IEEE 802.1X (EAPOL) [1] for port-based network access control being used especially in wireless networks by IEEE 802.11i (WPA2) [2], *SEcure Neighbor Discovery (SEND)* [3] for IPv6, and our *Cryptographic Link Layer (CLL)* [4], which provides authentication and confidentiality to the hosts of the LAN by safeguarding all layer 2 traffic including ARP and DHCP. However, since public key operations are very expensive in comparison to symmetric cryptographic primitives like block ciphers and hash functions, they may constitute a new target for DoS attacks. Especially on the link layer an attacker may freely take on different identities (i. e., sender addresses) and send to the victim host a flood of bogus connection requests each one requiring to perform an expensive public key operation. The victim host becomes overloaded and very likely cannot process the requests originating from benign hosts any more. Existing DoS protection schemes turn out to have drawbacks in environments with an eavesdropping attacker where initially no address authenticity exists.

In this paper we propose a countermeasure against DoS flooding attacks on public key handshakes in wired and wireless LANs. The idea is to configure on each host a reasonable threshold of signatures (or certificates) to be checked per second without overloading its CPU. A benign host trying to initiate an authentication handshake to a victim system being currently under a flooding attack reacts to this aggression by flooding itself multiple copies of its request packet for a short period. We call this approach *counter-flooding*. The attacked host collects all incoming request packets for a certain time interval and afterwards processes only those packets having the largest number of duplicates. Usually the benign host has to flood its request packet only for some tens of milliseconds to ensure that it will be definitely processed by the victim. The key point is that the adversary cannot deliver many duplicates without at the same time reducing the victim's workload, i. e., the number of distinct signatures to verify. We take advantage of broadcast transmissions to detect DoS flooding attacks and run our counter-flooding defense only when necessary. We show that our duplicate-based selection policy for incoming requests outperforms a probabilistic arbitration mechanism. The practical applicability of our approach is underlined through flooding experiments with different Ethernet switches. In addition we demonstrate how IEEE 802.3x flow control can protect from DoS flooding attacks attempting to significantly degrade TCP throughput due to excessive packet loss.

The remainder of this paper is organized as follows. In the next section, we discuss existing approaches on protecting hosts against resource depletion through a flood of bogus request packets. Section III presents the design of the counter-flooding mechanism. In Section IV we describe a queuing extension for our scheme and assess impact and quality of counter-flooding. Section V provides experimental results for bandwidth division during a flooding attack in switched Ethernet. Finally, we conclude the paper with a summary in Section VI.

II. Related Work

A comprehensive survey on DoS attacks and proposed defense mechanisms can be found in [5]. The authors classify four categories of defense: (1) attack prevention, (2) attack detection, (3) attack source identification, and (4) attack reaction. Our counter-flooding mechanism falls into the last category. It is a currency-based approach. The host under attack demands from its clients to pay in some currency—in our case bandwidth—before spending itself resources to process their incoming requests. In the following we take a closer look on existing currency-based DoS defense mechanisms and point out their shortcomings if applied in LAN environments.

In [6] Juels and Brainard introduced *client puzzles* to protect servers from TCP SYN flooding attacks. Being under attack, a server distributes to its clients cryptographic puzzles in a stateless manner asking them to reverse a one-way hash function by brute force. The difficulty of the puzzle is chosen depending on the attack strength. Only after receiving a correct solution from the client the server allocates resources for the dangling TCP connection. The idea of CPU-bound client puzzles has been applied to authentication protocols in general by Aura et al. in [7]. An implementation of client puzzles to protect the TLS handshake against DoS is described in [8]. However, the application of client puzzles itself may become the new target for an attacker if no address authenticity is provided by the underlying layers. The adversary can mount a second flooding attack against the clients of the defending host by overwhelming them with bogus puzzles pretending to come from the defending host. Depending on the chosen puzzle strength, even a modest puzzle packet rate may be sufficient to prevent the clients from solving the authentic puzzle set by the defending host. The ability to eavesdrop on the LAN traffic alleviates the puzzle attack since the attacker gets to know the clients performing currently an authentication handshake.

Martinovic et al. [9], [10] address DoS attacks in IEEE 802.11 networks aiming to exhaust the access point's (AP) resources by flooding it with fake authentication requests. In [9] they propose a scheme called *Early MAC Address Binding* to protect the IEEE 802.11i / 802.1X handshakes. It involves a Diffie-Hellman key exchange yielding a temporary message authentication code for the otherwise unsecured handshake frames and a cookie mechanism to bind the authentication requests to MAC addresses. However, this countermeasure only raises the bar for a successful DoS attack, but does not provide a complete protection. In [10] the authors suggest *wireless client puzzles* to be distributed by a defending AP to joining stations. To support highly heterogeneous stations these puzzles are not CPU-bound like in Juels' scheme. Instead of inverting a one-way function, a station has to measure the signal strength of the links to its neighbors and to find out those neighbors, whose link reaches a certain *Neighborhood Signal Threshold (NST)*. The NST is randomly chosen and frequently changed by the AP. A station replying with a wrong solution is detected by its neighbors, which thereupon issue a warning to the AP. However, similarly to the client puzzle attack described above, an adversary may impersonate the AP and announce many different NST values thus sabotaging the verification. The assumption that already authenticated stations are always benign and do not purposely issue false warnings may also not hold for some scenarios. In contrast, our counter-flooding mechanism does not require any trust. It would provide a viable protection for IEEE 802.11i / 802.1X handshakes.

Using bandwidth as a currency to defend against application-level distributed DoS attacks was proposed by Walfish et al. in [11]. The idea of their defense mechanism called *speak-up* is the following: a victimized server asks its clients to open a separate payment channel and to send through it some dummy bytes to the server. Each time the server is ready to process a new client request, it holds a virtual auction and selects the client that has sent the most bytes so far. At the same time the corresponding payment channel is terminated. While speak-up operates at the application layer and requires the establishment of TCP connections, our counter-flooding approach is more low level and deals with single packets. Speak-up implicitly relies on address authenticity to assign the payment, i. e., the received dummy bytes, to the corresponding client request. In the considered LAN scenario an (eavesdropping) attacker can break the speak-up mechanism by impersonating the benign clients and sending fake requests to the defending server which will likely spend the clients' payment to process the fake request instead of the genuine one. Unlike speak-up, our counter-flooding approach is resistant against this impersonation attack since a client pays by sending multiple copies of its request. Here the payment cannot be spent without processing the genuine request itself.

In [12] Gunter et al. proposed a broadcast authentication protocol with DoS protection and packet loss tolerance based on digital signatures. To defend against a flood of fake signature packets the receiver checks each incoming signature only with a certain probability while the benign host sends multiple copies of his signature packet to raise its chances on verification. However, the authors calculate the required number of duplicate packets for a benign sender to succeed with high probability by making the unrealistic assumption that all copies will arrive at the receiver contiguously in a single pile without being intermixed with the attacker's signature packets. In contrast, we do not assume any special packet order in our counter-flooding approach and provide bounds on the number of duplicates to deterministically guarantee the processing of the good packet at the receiver. Our

TABLE I
SYMMETRIC VS. ASYMMETRIC KEY CRYPTOGRAPHY.

operation	*speed*
HMAC (MD5)	362.4 MB/s [371 100 pkt/s]
HMAC (SHA-1)	290.2 MB/s [297 170 pkt/s]
AES	135.8 MB/s [139 060 pkt/s]
RSA-1024 verify	10 676 pkt/s
RSA-2048 verify	3663 pkt/s
RSA-3072 verify	1972 pkt/s
RSA-4096 verify	1135 pkt/s
DSA-1024 verify	946 pkt/s
DSA-2048 verify	312 pkt/s

duplicate-based arbitration scheme turns out to be more efficient than random checking.

III. DESIGN OF COUNTER-FLOODING

A. Goal: Safeguarding the Public Key Handshake

Table I compares the speed of symmetric versus asymmetric cryptography to authenticate 1 KB packets at the receiver. For the benchmark we used the popular crypto library OpenSSL on an Intel Core 2 Duo 2.66 GHz machine (one core active). The public exponent employed for RSA signatures was 65 537. Using a Hashed Message Authentication Code (HMAC), current computers can easily authenticate incoming packets at full link-speed in IEEE 802.11g wireless networks, in 100 Mbit and even in 1 Gbit Ethernet LANs. Symmetric encryption/decryption using a block cipher like AES is also quite fast. In contrast, depending on the chosen modulus size, the verification of RSA signatures may already become a bottleneck at link speeds far below the 100 Mbit mark. Checking DSA signatures is even slower. Note that in public key cryptography, the workload is primary determined by the number of packets and not by their size.

The benchmark results indicate that network communication protected by symmetric key cryptography (like message authentication code and block cipher) is in general not vulnerable to DoS flooding attacks, since the employed algorithms are usually fast enough to process packets at full link-speed. On the other hand the verification of digital signatures (or certificates) is quite slow. Thus we suppose that public key authentication handshakes, which are often performed in the setup phase before switching to symmetric key cryptography, may be sabotaged by flooding the victim host with connection request packets containing bogus signatures. Our goal is therefore to protect the computationally expensive public key handshake against DoS. In more general terms, we want to ensure the successful processing of important, infrequently sent genuine signature packets (be it handshake or status messages) despite of attackers flooding fake packets.

B. Basic Idea

We define for all hosts in the LAN a common threshold f for the maximum number of signature packets received per second during normal operation. f usually depends on the employed security protocol and the network size. If this threshold is exceeded, the receiver is considered to be under a DoS flooding attack and the counter-flooding protection mechanism gets invoked. Conversely, when the rate of incoming signature packets falls below the threshold f again, we return back to normal mode. In broadcast networks the benign sender A of a signature packet addressed to an attacked host B can overhear all the packets delivered to B and detect whether the threshold has been exceeded. In other networks, e. g., in switched Ethernet, one can instead prescribe that valid signature packets must always be sent to the broadcast MAC address to achieve the same effect. In this case A's MAC address must be stated in the packet's payload and A will drop any signature packet addressed directly to its own MAC address. A different approach for A would be to consider B as being attacked, if B does not reply to or acknowledge A's signature packet even after several retransmissions. However, in that case host A cannot exclude that there is no attack and B is simply currently offline.

When being under attack, host B divides the time into periods of length t (in the order of milliseconds) and collects all incoming signature packets in each period. Duplicate packets are stored only once. The queue maintains a duplicate counter for each packet stating how many times it has been received during the current period. Depending on B's CPU power (and its average load) the user configures a *verification threshold* of v signature checks per period t. This determines the maximum amount of CPU power B is willing to spend on the verification of signature packets. At the end of the current period all queued packets except for the v packets with the highest duplicate counter value are dropped. The remaining v packets have "won" the selection and their signatures will be verified during the next period. This verification can either be performed quickly at high CPU load straight at the beginning of the next period or it can be spread over the whole period length t.

We use bandwidth as a currency and request host A to flood its genuine signature packet addressed to victim B for a short time $p \leq t$ to ensure that it will definitely be processed. The intuition of counter-flooding is to combat the attacker using his own weapons by verifying only those packets, that have a large number of duplicates. In LANs we can usually assume that host A and the attacker have equal or at least similar bandwidth capabilities. Therefore, the saboteur cannot afford to deliver a large number of different fake signature packets while simultaneously maintaining for all these packets a duplicate counter value as high as the value of A's packet in victim B's queue. The higher the bandwidth of host A's counter-flooding measure, the shorter we can choose the period length t and the required flooding duration p, i. e., the faster A's signature packet will be processed. Like the attacker, host A will usually flood at full link speed.

C. Bandwidth vs. Packet Count

Host B's verification threshold v is expressed in the number of signature packets, while bandwidth is a measure for the number of bytes transferred per time unit. The attacker may try to gain an advantage over host A by flooding small signature packets without payload and—having the same link speed as A—hereby deliver more signature packets to victim B. For example, the length of an RSA-1024 signature is 128 bytes while the Maximum Transmit Unit (MTU) in Ethernet LANs usually is 1500 bytes (in Gigabit Ethernet even up to 9000 bytes). We address this issue by prescribing that valid signature packets must be padded with zeros to have maximum MTU size. This way all signature packets have equal size and we reduce the rate at which the attacker can deliver fake packets right from the start.

D. Determining the Flooding Duration

Now we determine the required flooding duration p for host A to guarantee that victim B will process its signature packet. Let b be the bandwidth of the links in the given network expressed in the number of maximum MTU packets that can be transmitted per second. In case of a switched Ethernet both A and the attacker can send packets addressed to B at full rate b simultaneously, but the switch will deliver only half of the packets. The rest will be dropped due to queue overflow. A fair arbitration mechanism which serves all input ports in equal shares in case of an overload condition at an output port is the design goal of every good switch [13]. Under ideal conditions victim B should therefore receive approximately the same number of packets from A and the attacker. We experimentally examine this aspect in Section V. When operating in a wireless environment all senders have to share the available bandwidth b. Assuming a (somewhat) fair medium access control [14] both A and the attacker should achieve approximately the same throughput and packet delivery rate with $\frac{b}{2}$ as the optimum if no other senders are active. However, our counter-flooding mechanism does not require a fair bandwidth division between host A and the attacker. Host A may send and deliver its signature packets at a lower rate than the attacker because of other packet flows dispatched by A or because of an unfair switch or medium access control. The only assumption we make is that host A can always achieve a packet delivery rate of at least $k \cdot b$ where $0 < k < 1$. That is, factor k denotes A's minimum link share with respect to signature packets addressed to B during the counter-flooding action.

Having the downlink bandwidth b victim B may receive up to $b \cdot t$ signature packets during a period of length t where at least $k \cdot b \cdot p$ duplicate packets originate from host A. From all received signature packets only v packets having the highest duplicate counters will be selected for verification. In total there can be no more than $\frac{b\,t}{k\,b\,p}$ packets having a duplicate counter equal or greater to the number of duplicates of A's packet. Thus if the inequality

$$\frac{b\,t}{k\,b\,p} \leq v \tag{1}$$

holds, host A's signature packet will be definitely among the v packets selected for verification during the next period. For the flooding duration $p \leq t$ we now get

$$p \geq \frac{t}{k\,v} \quad \Leftrightarrow \quad \frac{p}{t} \geq \frac{1}{k\,v} \tag{2}$$

where the condition $k\,v \geq 1$ must be fulfilled.

So far we have implicitly assumed that the hosts A and B are time synchronized and A exactly knows when for B a new period begins. But a time synchronization between the hosts in the range of milliseconds is usually unrealistic. We address this issue by extending the flooding duration p to $p' = 2p$ and eliminate the dependency on time synchronization. This way host A definitely hits the beginning of a new period t and contributes to it with its duplicates for a time span of at least p.

E. Choosing the Parameters

The first step is to estimate the factor k. In case of a single attacker, one host performing counter-flooding, and a fair bandwidth division between the two k is 0.5. However, there can be multiple attackers (say, up to g) and/or several hosts (say, up to h) performing counter-flooding simultaneously for the same victim host. In this general case, assuming a fair bandwidth division, k can be set to $\frac{1}{g+h}$. To address possible unfairness in the bandwidth division we now introduce an *unbalance factor* u, $0 < u \leq 1$, and propose for k the expression $\frac{u}{g+h}$. The unbalance can be attributed to a suboptimal switch behavior or medium access control but also to the uplink of host A being concurrently used by some other packet flows. A reasonable value for u may be, for example, 0.5 or 0.3. The value for g depends on the network size and the (empiric) threat level, while h is primary determined by the number of hosts sending a signature packet to the same destination and the frequency of these packets.

So far we have not considered the impact of other packet flows occupying bandwidth on the victim's downlink. In a switched Ethernet these are all the other packets addressed to victim B, while in a wireless environment the remaining packet transmissions of the whole network altogether reduce the victim's downlink bandwidth available for signature packets. However, since the flooding duration p does not depend on the link bandwidth b, we can still use our model if we assume that b is not the physical bandwidth of the network links, but the bandwidth of the victim's downlink currently occupied by the flooded signature packets.

Now that we know k, we select a global value for v such that the inequality $k\,v \geq 1$ is fulfilled. We propose to set $v_{glob} = \frac{4}{k}$ yielding $\frac{p}{t} \geq \frac{1}{4}$ to have some latitude

when choosing p. The last step is to select t large enough so that the majority of hosts in the network can verify v_{glob} signature packets in t without getting overloaded. Each user can now configure on its machine a local value for v, which may be of course larger (fast CPU) but also smaller (slow CPU) than the global value. In the first case the victim host will be robust against an overestimate of k, i. e., against a more powerful attack. In the second case it may fail to process host A's signature packet. But we can deal with hosts having a low v_{loc} value by increasing the flooding duration p up to the period length t. If victim B does not reply to or acknowledge A's signature packet, host A can perform counter-flooding repeatedly doubling each time the value for p in case of failure. Having $v_{glob} = \frac{4}{k}$ and $v_{loc} \geq \frac{1}{4} v_{glob}$ host A needs to double p at most twice (yielding $p = t$) to guarantee that victim B will definitely process its signature packet. Without time synchronization the real flooding duration remains, of course, $p' = 2p$.

To give an impression for the practical feasibility of the counter-flooding mechanism in an office LAN consisting of, e. g., 100 off-the-shelf computers we now calculate the real flooding duration p' using the following values: up to $g = 3$ attackers, up to $h = 2$ hosts performing counter-flooding simultaneously for the same victim, unbalance factor $u = 0.4$, $k = \frac{u}{g+h} = 0.08$, $v = \frac{4}{k} = 50$, verifying RSA-2048 signatures on a 2.66 GHz CPU (see Table I) with an average CPU load of 25% $\Rightarrow t = \frac{50}{0.25 \cdot 3663} = 54.6$ ms. This yields $p \geq \frac{t}{k\,v} = 13.7$ ms and $p' = 2p \geq 27.4$ ms. Counter-flooding ensures that in case of a DoS attack victim B will process host A's signature packet with a delay of less than $p + 2t = 122.9$ ms (hitting the beginning of a new time period t, collecting packets during a full period, and verifying v signatures during the next period).

IV. More Details

A. Reducing the Queue Size

During the period t the victim host needs to queue all incoming *different* signature packets. Depending on the link bandwidth b the memory requirement might be an issue for low-end and embedded systems. However, since all valid signature packets are padded to have maximum MTU size, only the meaningful part of the packet has to be stored. Furthermore we propose to significantly reduce the required memory footprint by employing a cryptographic hash function. The idea is to keep only v full packets—those having currently the largest duplicate counter—in the queue, while all other packets are represented by their hash value. A digest size of 64 or 80 bits should be large enough to avoid collisions. Then only 8 or 10 bytes (instead of, e. g., 1000 bytes) are required per packet. When more duplicates of a packet, which is right now represented in the queue only by its digest, arrive, it gets enqueued as a full packet and takes up the place of one of the v packets having a lower duplicate counter. The ousted packet is replaced by its digest. The described algorithm ensures that the v packets with the largest duplicate counters are definitely available for verification at the end of period t.

B. Impact of Counter-Flooding on Network Performance

We believe that the impact of counter-flooding on the overall network performance is negligible. Our DoS countermeasure is very goal-oriented. It comes into action only in case of a real flooding attack, only when performing an authentication handshake, and only for a very short period of time in the order of 20–60 ms. Ordinary file transfer usually occupies a large portion of bandwidth for much longer periods. Another point is that by flooding *broadcast* packets at full speed (or generally in wireless networks) a single attacker can already induce an overload situation for the whole network. In this case counter-flooding cannot increase the network load any more since it has already reached the maximum. Here the bandwidth occupied by counter-flooding comes at the cost of the attacker's bandwidth.

C. Comparison to a Probabilistic Arbitration Scheme

Counter-flooding employs a deterministic arbitration scheme at victim B, which selects the signature packets with the highest number of duplicates for verification. We now compare this scheme to a probabilistic one inspired by [12], in which from the n packets (including duplicates) received during period t the victim host *randomly* chooses v packets for verification. Let in this case m be the number of packets originating from host A. Then, using simple combinatorics, the probability that at least one of A's packets will be among the v selected ones can be expressed by

$$\beta = \frac{\sum_{i=0}^{v-1} \binom{m}{v-i} \binom{n-m}{i}}{\binom{n}{v}}. \tag{3}$$

Taking the parameters $k = 0.08$, $v = 50$, $t = 54.6$ ms and $p = 13.7$ ms from the example in Section III-E, we compute now the probability β in a 100 Mbit and 1 Gbit Ethernet LAN. Assuming a frame size of 1518 bytes at the link layer, we get $n = 450,\ m = 9$ for $b = 100$ Mbit/s and use $n = 4500,\ m = 90$ for $b = 1$ Gbit/s. This yields a probability β of only 65.7% (100 Mbit) and 63.8% (1 Gbit) respectively. When maximizing m by setting $p = t$ (i. e., quadrupling the flooding duration) the verification probability β reaches a value of 98.8% and 98.5% respectively.

This comparison demonstrates that our deterministic duplicate-counter-based selection policy clearly outperforms the probabilistic one. In the probabilistic approach the chances for verification decrease when the bandwidth

TABLE II
BANDWIDTH DIVISION DURING COUNTER-FLOODING UNDER DIFFERENT CONDITIONS.

conditions	***host A's bandwidth share k´***					
	LevelOne FSW-2205TX 100 Mbit		*3com OfficeConnect 8 Plus 100 Mbit*		*3Com OfficeConnect 5 1 Gbit*	
	$t = 50\ ms$	$t = 80\ ms$	$t = 50\ ms$	$t = 80\ ms$	$t = 50\ ms$	$t = 80\ ms$
CF_U noFC	47.8% – 51.3%	49.1% – 51.4%	33.5% – 60.7%	33.4% – 66.9%	—	—
CF_U FC	52.5% – 54.9%	52.4% – 52.5%	95.8% – 99.7%	97.5% – 100%	49.7% – 49.9%	49.6 – 49.7%
CF_B noFC	47.5% – 52.5%	47.7% – 52.5%	49.9% – 50.0%	50.0% – 50.1%	—	—
CF_B FC	55.0% – 55.2%	54.1% – 55.0%	46.7% – 50.1%	50.0% – 50.1%	49.9% – 50.1%	50.1% – 50.6%
CF_U TCP noFC	19.1% – 23.6%	19.2% – 24.3%	20.9% – 37.1%	27.7% – 37.2%	—	—
CF_U TCP FC	23.2% – 27.8%	21.9% – 27.9%	28.5% – 36.6%	27.8% – 36.9%	27.1% – 33.9%	26.3% – 33.4%
CF_B TCP noFC	38.6% – 50.1%	40.0% – 52.1%	49.6% – 50.3%	49.8% – 53.6%	—	—
CF_B TCP FC	26.1% – 31.5%	25.3% – 29.4%	28.3% – 40.2%	29.4% – 38.7%	28.4% – 38.5%	28.6% – 37.6%

increases, while the deterministic strategy does not depend on the link speed. The advantage of the duplicate-counter-based strategy can be explained as follows: the verification of one forged signature packet eliminates the need to check a multiplicity of other forged packets, namely all its duplicates, while each verification of a randomly selected packet excludes only this single packet from the candidate queue.

V. FLOODING EXPERIMENTS IN SWITCHED ETHERNET

In this section we present the results from flooding experiments with Ethernet switches to examine the bandwidth division between concurrent packet flows.

A. IEEE 802.3x Flow Control

To address the problem of switch congestion resulting in packet loss, the IEEE 802.3x Task Force specified for full duplex Ethernet a hop-by-hop flow control scheme—the PAUSE function [13], [15]. Whenever the switch receives packets faster than they can be forwarded to the output port(s), it emits a special PAUSE frame at the corresponding input port(s). The PAUSE frame asks the station at the other end of the link to stop transmitting further packets for a specified amount of time. This enables the switch to empty its input buffer without discarding packets. Both the switch and the station's NIC must support flow control to take advantage of it. This is the case for the majority of today's hardware. However, flow control often needs to be activated manually.

Dealing with flooding attacks which usually provoke an overload condition at the switch, it seems important to take flow control into account for our experiments.

B. Bandwidth Division between Host A and Attacker

We examine the fairness of bandwidth division between the benign host A and the attacker when performing counter-flooding. This helps to properly choose the factor k. Using in turn three different switches we set up a LAN consisting of four hosts: the attacker, victim B, host A and another host C. The hosts A, B, and C are notebooks equipped with an Intel Core 2 Duo 1.60 GHz CPU and an Intel 8256MM Gigabit NIC, while the attacker's machine has an Intel Core 2 Duo 2.66 GHz CPU with a Realtek 8111B Gigabit NIC. The first two switches, a LevelOne FSW-2205TX and a 3com OfficeConnect 8 Plus, operate at 100 Mbit. The third one, a 3Com OfficeConnect 5, provides Gigabit speed. Injecting and capturing raw Ethernet frames is performed by means of the *pcap* library. The attacker continuously floods maximum MTU packets at full link-speed addressed to victim B, which registers all packets received during time periods of length $t = 50$ ms in the first and $t = 80$ ms in the second measurement. Host A reacts to this aggression by counter-flooding maximum MTU packets for $p' = 2\,t$ to definitely hit a full period t at B. We record the number of packets received from A and the attacker during the second period of length t with packet contribution from both hosts. This is the desired full period with A's packets. The number of A's packets divided by the total number of packets yields its bandwidth share k'. We measure k' under various conditions: counter-flooding unicast (CF_U) or broadcast (CF_B) packets, with flow control disabled (noFC) or enabled (FC), and having the entire uplink bandwidth available or running in parallel to a TCP connection which tries to send data to host C at maximum speed. The attacker always operates with flow control disabled. However, he adheres to the policy whether valid signature packets are broadcasted or unicasted. Otherwise his fake packets can be detected based on the destination MAC address. For each configuration we perform 15 runs and state the ranges of the measured values for k' in Table II. In Gigabit Ethernet switching off flow control has no effect in case of the two employed NIC chipsets—in remains in fact enabled. Therefore there are no test results without flow control for the Gigabit switch.

Evaluating the results, the general observation to be made is that host A's bandwidth share did never drop below 33% (entire uplink channel available) and 19% (concurrent TCP flow) respectively. Hence, the bandwidth division can be assessed as reasonably fair and, more important, an acceptably large lower bound for factor k exists. Using the expression $k = \frac{u}{g+h}$ proposed in Section III-E with $g = 1$ and $h = 1$, the unbalance factor u yielding $k = 0.19$ would amount to 0.38.

Without a concurrent TCP flow, when using the first or the third switch, the bandwidth division between host A and the attacker is almost optimal, that is $k' \approx 50\%$, in all runs. Flow control has no significant effect on fairness

TABLE III
EFFECT OF ETHERNET FLOW CONTROL ON TCP THROUGHPUT UNDER A DOS FLOODING ATTACK.

conditions		*TCP throughput in MB/s*		
		switch 1	switch 2	switch 3
noFC	noDoS	11.32 – 11.33	11.32 – 11.33	—
FC	noDoS	11.32 – 11.33	11.32 – 11.33	102.7 – 110.1
noFC	DoS_U	0.294 – 0.607	0.117 – 0.138	—
FC	DoS_U	5.91 – 5.94	10.47 – 10.63	52.94 – 53.16
FC	DoS_B	5.93 – 5.96	10.54 – 10.56	44.29 – 44.62

in case of the first switch. However, the second switch shows striking fluctuations in bandwidth division when flooding unicast packets without flow control and it drops almost all attacker's unicast packets when A has flow control enabled. That is, this switch favors stations which perform flow control. All test series with a concurrent TCP flow exhibit medium-strength fluctuations of A's bandwidth share. This is probably due to varying TCP throughput occupying more or less uplink bandwidth and maybe also due to short-time monopolization of A's uplink either by the TCP or the counter-flooding flow. If TCP congestion control does not throttle the sending rate, a fair use of A's uplink by TCP and counter-flooding as well as a fair arbitration mechanism in the switch would result in k' to amount to 25%. It is noticeable that in the broadcast test series without flow control k' is between 40% and 50%. We explain this by the fact that persistent packet loss in the overloaded switch forces TCP's congestion control to dramatically cut down its throughput, so that counter-flooding can occupy nearly the whole uplink. The fatal effect of flooding attacks on TCP throughput is subject of the next subsection.

C. Preventing DoS Flooding Attacks on TCP

Using the existing setup, we let the attacker flood at full speed unicast or broadcast packets of maximum size to victim B thereby completely occupying its downlink. While the attack is in progress, host A runs the TCP benchmark tool *ttcp* sending data to victim B as fast as possible (40 KB buffer, 500 *send()* calls). We examine the impact of Ethernet flow control on the achievable TCP throughput by making measurements with and without flow control enabled at the hosts A and B. The attacker has flow control always switched off. Another test series is performed under normal conditions without a flooding attack to determine the baseline TCP throughput. For each configuration we perform 15 runs and present the ranges of the measured speeds in Table III.

Under normal conditions the TCP throughput achieves (almost) link-speed irrespective of whether IEEE 802.3x flow control is enabled or not. Without flow control the flooding attack succeeds in degrading TCP throughput by 95–99% due to packet loss which continuously triggers TCP's slow start mode. This constitutes a very severe DoS attack since the majority of applications employ TCP. However, Ethernet flow control is capable to maintain a TCP throughput of more than 50% of the regular speed, as it avoids packet loss. We believe that a link layer security protocol implementing the counter-flooding protection for its public key handshake combined with this finding paves the way for safe and DoS-resistant communication in LANs.

VI. CONCLUSION

In this paper we have introduced a countermeasure against DoS flooding attacks on public key handshakes. If an adversary tries to overwhelm a victim host by a flood of invalid signature packets requiring expensive verification, a benign host ensures the processing of its genuine signature packet by flooding itself copies of this packet for a short period. We have provided bounds for counter-flooding to succeed and shown experimentally that in switched Ethernet a reasonable fair bandwidth division between concurrent flows is usually ensured.

REFERENCES

[1] IEEE 802.1X-2004, "Port Based Network Access Control," IEEE Standard, Dec. 2004.

[2] IEEE 802.11i-2004, "Amendment 6: Medium Access Control (MAC) Security Enhancements," IEEE Standard, Jul. 2004.

[3] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," RFC 3971, Mar. 2005.

[4] Y. I. Jerschow, C. Lochert, B. Scheuermann, and M. Mauve, "CLL: A Cryptographic Link Layer for Local Area Networks," in *SCN 2008: Proceedings of the 6th Conference on Security and Cryptography for Networks*, Sep. 2008, pp. 21–38.

[5] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computing Surveys*, vol. 39, no. 1, p. 3, 2007.

[6] A. Juels and J. G. Brainard, "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks," in *NDSS '99: Proceedings of the Network and Distributed System Security Symposium*, Feb. 1999.

[7] T. Aura, P. Nikander, and J. Leiwo, "DOS-Resistant Authentication with Client Puzzles," in *Revised Papers from the 8th International Workshop on Security Protocols*, Apr. 2001, pp. 170–177.

[8] D. Dean and A. Stubblefield, "Using client puzzles to protect TLS," in *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, Aug. 2001, pp. 1–1.

[9] I. Martinovic, F. A. Zdarsky, and J. B. Schmitt, "On the Way to IEEE 802.11 DoS Resilience," in *Proceedings of IFIP NETWORKING 2006, Workshop on Security and Privacy in Mobile and Wireless Networking*, May 2006.

[10] I. Martinovic, F. A. Zdarsky, M. Wilhelm, C. Wegmann, and J. B. Schmitt, "Wireless Client Puzzles in IEEE 802.11 Networks: Security by Wireless," in *Proceedings of the ACM Conference on Wireless Network Security (WiSec 2008)*, Mar. 2008.

[11] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS defense by offense," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, Sep. 2006, pp. 303–314.

[12] C. A. Gunter, S. Khanna, K. Tan, and S. S. Venkatesh, "DoS Protection for Reliably Authenticated Broadcast," in *NDSS '04: Proceedings of the Network and Distributed System Security Symposium*, Feb. 2004.

[13] R. Seifert, *The Switch book*. Wiley, 2000.

[14] B. Bensaou, Y. Wang, and C. C. Ko, "Fair medium access in 802.11 based wireless ad-hoc networks," in *MobiHoc '00: Proceedings of the 1st ACM Interational Symposium on Mobile Ad Hoc Networking and Computing*, Aug. 2000, pp. 99–106.

[15] IEEE 802.3-2005, "Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications, Annex 31B," IEEE Standard, Dec. 2005.