DoS Protection for Reliably Authenticated Broadcast*

Carl A. Gunter, Sanjeev Khanna, Kaijun Tan, and Santosh Venkatesh
University of Pennsylvania

Abstract

Authenticating broadcast packet communications poses a challenge that cannot be addressed efficiently with public key signatures on each packet, or securely with the use of a pre-distributed shared secret key, or practically with unicast tunnels. Unreliability is an intrinsic problem: many broadcast protocols assume that some information will be lost, making it problematic to amortize the cost of a single public key signature across multiple packets. Forward Error Correction (FEC) can compensate for loss of packets, but denial of service risks prevent the naive use of both public keys and FEC in authentication. In this paper we introduce a protocol, *Broadcast Authentication Streams (BAS)*, that overcomes these barriers and provides a simple and efficient scheme for authenticating broadcast packet communications based on a new technique called *selective verification*. We analyze BAS theoretically, experimentally, and architecturally.

1 Introduction

Authenticating packet broadcasts is an interesting technical challenge. If a common symmetric key is used to authenticate the packets then any party that has the key can spoof the broadcast. Using a public key signature would solve this problem, but signing and verifying each packet would be expensive. Signing a group of packets would reduce this cost, but many types of broadcast are unreliable: if a signature is for a group of packets and one or more of them is dropped, then it is not possible to check the signature. A variety of ideas have been proposed to address this problem with public keys by including some kind of redundancy such as including multiple copies of hash values or using Forward Error Correction (FEC). Such approaches are vulnerable to a Denial of Service (DoS) attack on either the redundancy scheme itself or the public key signature packets. In the former case, false parity information makes it expensive to reconstruct valid packets, and, in the latter case, a flood of false signatures burdens the recipient with too many signature checks.

This paper proposes an approach to solving these and other DoS threats by a technique called *selective verification*. The idea is to use public key signature packets to authenticate hashes, parity information, and data, while using random checking of signature packets to defend against signature flooding and sequence numbers and time stamps to combat replay attacks. The sender sends many copies of its signature packets and the recipient checks the signature packets it receives with a given probability. The number of copies and the probability of verification can be varied to match the load that the recipient is able to check to cope with an expected level of attack. For example, suppose a sender sends a 10Mbps stream to a receiver, but this is mixed with a 10Mbps stream of DoS packets devoted entirely to bad signatures. To relieve the recipient of the need to check all of these bad signatures, the receiver can check signatures with a probability of 25%, and, if the sender sends about 20 copies of each signature packet, the receiver will find a valid packet with a probability of more than 99% even if the network drops 40% of the sender's packets. This technique is inexpensive, effective against severe DoS attacks, and adaptable to many different network characteristics.

To demonstrate selective verification we have developed a protocol called Broadcast Authentication Streams (BAS) that provides authentication for a data stream by adding a pair of authentication streams. Our target is to show that BAS can be used effectively on stock PCs over mid to high bandwidth links ranging from 10Mbps to almost 1Gbps. To this end we provide a theoretical analysis of BAS, an implementation, and experiments. The theory describes how to set parameters that affect features such as latency, overhead, recovery confidence, and maximum buffer size requirements. The experiments confirm our theoretical calculations and provide practical information such as throughputs on the target systems. Our implementation of BAS for a sender takes an array of input RTP packets and produces an array of authentication data. For a receiver, it takes an array of packets based on models of packet loss and a "shared channel" model

*In Network and Distributed System Security (NDSS), San Diego, California, February 2004.

of DoS attacks introduced in this paper and processes these to find authenticated data packets. These experiments confirm estimates like the one above and provide information about throughput capacities. For instance, the packets from a 10Mbps sender with 40% loss can be authenticated with the use of less than 10% of processor time even during an attack of 10Mbps. Our theory predicts and our experiments confirm that selective verification and BAS work well on PCs for applications that can tolerate 1-2 second latencies under signature flood attacks that range up to 500Mbps. This can be compared to latencies for playout buffers for Internet multi-media streams, which often use latencies of 5-10 seconds.

Another contribution of the paper is a rigorous model for analysing and quantifying the effectiveness of a DoS protection scheme. We call the one introduced here the *shared channel model*. It is based on the idea that an adversary can insert packets into a valid stream and may affect the loss rate of the stream statistically, but cannot choose exactly which valid packets are actually received. This contrasts with the Dolev-Yao model in which the adversary is considered to have control over exactly which packets are delivered. We argue that the weaker model is more appropriate for analyzing DoS threats in many cases.

The paper is divided into nine sections. Section 2 introduces the shared channel model, which serves as the foundation for our analysis. Section 3 provides an informal description of the BAS protocol. Section 4 describes related work. We specify the BAS protocol in Section 5 and analyze it theoretically in Section 6. We describe our prototype implementation of BAS with selective verification in Section 7. This was used to carry out experiments described in Section 8. Section 9 concludes. Appendices contain details of the loss models and error-correction codes used, theoretical results and proofs, and additional experimental data.

2 Shared Channel Model

The *shared channel model* assumes that a legitimate sender and an attacker share a packet communication channel to a receiver. A given model is characterized by a 4-tuple (W_0, W_1, A, p) consisting of the minimum bandwidth W_0 of the sender, the maximum bandwidth W_1 of the sender (where $W_0 \leq W_1$), the bandwidth A of the adversary, and the loss rate p of the sender where $0 \leq p < 1$. The ratio $R = A/W_1$ is the *attack factor* of the model. When $R = 1$, this is a *proportionate* attack and, when $R > 1$, it is a *disproportionate* attack. A proportionate attack with a loss rate of 20% is depicted in Figure 1. In an *informed* shared channel model, the

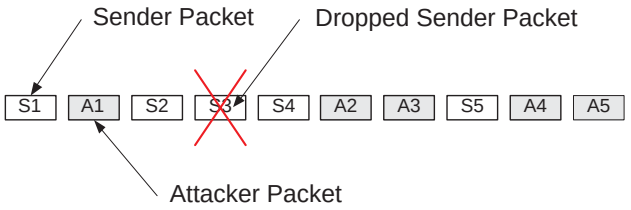

Figure 1: Shared Channel Model

adversary is assumed to know all of the values sent by the receiver and is able to forge predictable values like sequence numbers. This makes a considerable difference for some protocols. For instance, an informed attack on TCP could use sequence numbers in an ongoing connection to break the connection, providing a much cheaper attack than an uninformed approach like SYN flooding. An adversary may 'modify' a packet by replaying a modified version of a previously-seen sender packet. However, the adversary is unable to cause specific sender packets to be dropped or modified: sender packets are dropped probabilistically at rate p. The model assumes that sender packets arrive in the order in which they were sent when they do arrive. However, packets may appear to arrive out-of-order if an adversary replays dropped packets. Many protocols, including BAS, will treat reordered packets as dropped if they are grossly out-of-order (say, by more than a few hundred packets), so the assumption of in-order delivery for sender packets is not as strong as it may appear. Reordering can be modeled by taking a higher value of p and assuming that the adversary replays many dropped packets.

The shared channel model is more appropriate for analyzing denial of service than the much-studied Dolev-Yao model [3], which assumes that an adversary is able to drop all sender packets. An adversary with this ability is ensured of a DoS capability. The shared channel model is weaker than one that assumes that the adversary has all of the channel bandwidth ($W_1 = 0$). When a host can handle the load with this assumption (for instance, by rejecting all of bad packets without excessive processing), it can handle it with, say, a proportionate attack. But, there are cases (as shown later) where $W_0 \neq 0$ makes a major difference in how large A must be to achieve an effective attack. That is, if the legitimate sender can get *some* packets through, then these can be used to raise the bar for a successful DoS attack.

A *signature flood* attack is one in which an adversary sends false signatures. Checking these signatures is costly and burdens the victim's processor. Typical

Table 1: Cryptographic Costs

Crypto Operations	Operating Time
Sign(3)	4.92ms
Check(3)	86.8μs
Sign(17)	4.96ms
Check(17)	124μs
Sign(65,537)	5.10ms
Check(65,537)	270μs
Hash(1460)	13.3μs
Hash(10)	1.57μs

costs[1] are given in Table 1. The table provides signature and verification times for RSA on 10 bytes with exponents of 3, 17, and 65537 as well as SHA hashes on 10 and 1460 bytes. One can see from the figures for public key signatures that the processor is only able to create about 200 signatures each second. The costs of a Public Key Check (PKC) depends somewhat on the exponent. We will work with an exponent of 17 in this paper. A signature flood that results in 8000 PKC/sec would completely overwhelm a processor. We will typically work in terms of a PKC budget, for instance, one that requires that no more than 5% of processor time be spent on PKCs. So an effective flood could be achieved by forcing 400 PKC/sec. In a proportionate attack at high bandwidths, this could be easy. For instance, in Figure 1, if packet S5 holds a signature on hashes for packets S1-S4 and packets A1-A5 are false signature packets that look like S5 but contain a bad signature, then the receiver may end up checking most or all of these bad signatures. Since the entire bandwidth of the adversary could be devoted to the signature flood, the attack could realize a very high PKC burden at the receiver.

3 Informal Description

Our protocol, BAS, is based on a combination of FEC and repetition coding to provide modest bandwidth overhead and robust DoS protection. BAS uses FEC to reduce the costs of repeated hashes; it uses repeated signatures to secure the FEC-encoded hashes and address DoS attacks based on fake signature packets. Overall the architecture consists of three streams of packets as pictured in Figure 2. The first stream, called the *data stream*, consists of the data packets; these are not required to contain any cryptographic information.

[1]These are for a 2.4GHz PC with Redhat Linux 7.3 using operations from OpenSSL 0.9.6.

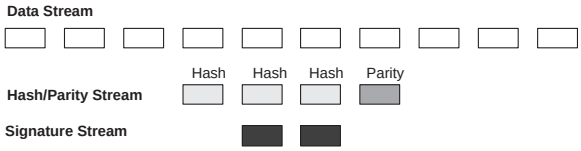

Figure 2: Architecture for Broadcast Authentication Streams (BAS)

The BAS protocol aims to authenticate this stream; the data packets may be encrypted if it is important to preserve their confidentiality, but the BAS protocol does provide this service. The second stream, called the *Hash/Parity (HP) stream*, consists of two kinds of packets. The first kind are called *hash* packets. These contain hashes of data packets. The second kind are called *parity* packets. These contain FEC coding information that allows dropped hash packets to be reconstructed. The third stream, called the *signature stream*, consists of packets that sign hashes of HP packets. The collection consisting of the data packets together with their corresponding hash, parity, and signature packets is called a *transmission group (TG)*. Figure 2 shows the packets in one transmission group. In general, a transmission group will include more than a thousand data packets, as determined by the allowable latency for authentication, a number of hash and parity packets that depends on the number of data packets and loss rate, and a number of signature packets that depends on the bandwidth and DoS threat.

Our approach to DoS prevention is based on two strategies. First, FEC-encoded information is protected by a digital signature. Thus spurious packets intended to burden FEC decoding will be discarded as quickly as their hashes can be checked. Second, we address DoS based on public key signature flooding with selective verification. We discuss two kinds of selective verification, *sequential* and *bin*. In both cases the idea is to send copies of signature packets to the receiver. The receiver checks a subset of the signature packets it receives. Some of these may be spoofed by an attacker, but the receiver will find a valid one quickly enough and with sufficiently modest computational effort to defeat signature flooding. In sequential verification, the receiver verifies signature packets randomly until finding a valid signature. After a sufficient number of such trials the probability of finding a valid signature will be high and the next signature packet can be sought. In bin selection we use sequence numbers in redundant signature packets. That is, the same signature is sent, but with a different sequence number for each copy. Suppose for example that a channel

with a 25% average loss rate between the sender and receiver supports an attack in which an adversary can send 500 fake signature packets per second. In bin verification, we divide the spoofing efforts of the adversary between a collection of signature packets using distinct sequence numbers. For example, suppose the sender creates 10 signature packets numbered 1 through 10 in a given second. The receiver waits long enough to receive some or all of these, together with up to 500 spoofed signature packets from an attacker. For at least 2 of the ten sequence numbers there will be no more than 102 spoofed and legitimate packets using that number. There is about a 93% probability that a legitimately signed packet is in this group of 102 packets. Thus the attacker is typically only able to force an additional 100 verifications with 500 spoofed packets. Increasing the number of sequence-numbered packets improves the burden at the receiver at the cost of a modest additional bandwidth.

Our approach to loss is also based on two strategies. The first, as discussed already, is to use FEC where possible and repetition where necessary. Since repetition was necessary to thwart denial of service in some cases anyway, and FEC offers very significant savings over repetition, this provides a comfortable tradeoff. The second strategy involves spacing hash, parity, and signature packets though the data stream to improve robustness against bursts. The idea is illustrated in Figure 3. The figure shows packets from three trans-

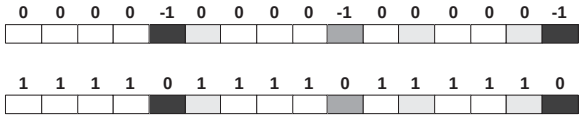

Figure 3: Interleaving of Streams

mission groups numbered $-1, 0, 1$ and having a profile similar to the TGs in Figure 2. In general, hash packets are sent as soon as they are ready while signature and parity packets are spaced throughout the data and hash packets of the subsequent transmission group. This interleaving adds robustness to burst loss at the potential cost of some additional latency.

4 Related Work

There are essentially three approaches to authenticating broadcast packets. The one we consider in this paper is to use 'ordinary' public key signatures to secure data, hash, and parity information. An alternative is to use public key signatures on the data and assure that the signature can be checked by using FEC to achieve reliability. A final alternative is to use symmetric keys or special kinds of public key signatures. Each of these approaches offers its own challenges. The first approach is vulnerable to signature floods, the second is vulnerable to attacks on FEC encoding as well as signature flooding, and the third requires additional assumptions such as synchronized clocks or special cryptography.

The idea of amortizing the cost of public key signing and checking over multiple packets by using hash chains was suggested in [5]. To deal with packet loss in authenticated broadcast, one approach that has been extensively studied is to add more hashes into the packets [18, 13, 6, 17]. For instance, a packet might contain the hash of the packet before it and the packet before that one. If the middle packet is lost, then the extra hash enables verification. This chaining has some problems, however. First, the choice of the chaining has a significant impact on the reliability that is achieved, and this impact is subtle to analyze. Second, reliability costs bandwidth since multiple hashes are needed to assure that chains to signature packets remain unbroken by losses. In any case, adding extra hashes to packets does not deal with dropped signature packets so something needs to be done about this too. A hybrid approach [24] addresses computation and reliability problems by putting a signature into each packet along with hashes of a collection of other packets. The signature is applied to a Merkle hash tree [12] created from this collection and only needs to be checked once for each collection. This also has the advantage that each packet can be verified as soon as it arrives. Despite the optimization provided by the Merkle tree, this approach has a significant bandwidth overhead since space must be allocated for the signature and multiple hashes in each packet.

The use of coding techniques to limit overhead for authenticated broadcast appears in the extended hash chain scheme of [18] based on Rabin's Independent Dispersal Algorithm (IDA). Subsequent work on FEC for authenticated broadcast has shown that low overheads can be acheived [14, 15]. These approaches are vulnerable to signature flooding and attacks on FEC encoding. An FEC-based technique to a address a DoS attack in which an adversary is able to *modify* a small number of sender packets is discussed in [15]. This technique is not effective when large numbers of invalid packets can be inserted by an adversary as in the shared channel model. For instance, suppose we wish to endure a proportionate attack with a code of n segments that can be reconstructed if no more than t segments are lost and no more than α are modified. Decoding is the first step in the verification phase of [15], so the

receiver will need to look for segments to use in decoding from among the segments he receives. Assume that $m \geq n - t$ of these came as sent by the sender and another n segments arrive as modified by an adversary. To decode, the receiver needs to find a set of $n - t$ segments from among $n + m$ possibilities that includes at most α segments from among the n adversary-modified segments. For reasonable values of n, t, α, m, testing these combinations until a good one is found will add many multiples of processing overhead to the decoding. This problem is the basis of the conventional wisdom [19, 9] that unauthenticated FEC-encoded data is vulnerable to DoS attack.

There are several approaches to broadcast authentication based on symmetric keys or special kinds of public key signatures. One approach [1] involves selecting a collection G of symmetric keys and distributing a random subset of G to each receiver; a trusted sender uses MACs based on all of the keys in G, and each receiver verifies using the keys they have. Another approach [18] assumes that the sender and receivers can synchronize their clocks to within an accpetable threshold. The sender then sends out data packets with MACs and later discloses the key for computing the MAC. Packets received after the disclosure are discarded. The use of MACs helps defeat DoS attacks becuase of their relatively low processing cost. When distributing sets of secret keys or synchronizing clocks is feasible these approaches are robust against DoS attacks. Of course, attention must then be paid to DoS treats against supporting protocols for time synchronization and key distribution. Another strategy to deal with DoS is to use less expensive forms of cryptographic verification such as one-time [5, 16] or k-time [21] signature schemes.

Our contribution in this paper is a technique, selective verification, and its application to DoS attacks on broadcast authentication schemes that use common kinds of public key signatures. Selective verification can be used with most of the existing approaches to public key authentication of broadcast packets. Our specific instantiation, the BAS protocol, uses a simple kind of hash tree divided into two authentication streams. This approach fits well with our use of RTP [23] and FEC techniques [20] that add parity packets as a supplementary stream (see also [4]). Our system achieves the low overheads of FEC-based techniques without the need for new kinds of cryptography or other assumptions and provides robust protection against DoS attacks.

5 Protocol Specification

As discussed broadly earlier, in order for broadcasts to be reliably authenticated, our protocol will provide two fundamental types of protection: (1) a mechanism enabling reliable recovery of authentication packets (*i.e.*, hash and signature packets) lost at *a priori* unpredictable locations; and (2) a configuration resilient to informed DoS attacks that provides computationally efficient authentication. We consider each of these issues in turn before providing a formal specification and analysis of the broadcast authentication streams (BAS) protocol.

5.1 FEC in the Authentication Stream

We refer the reader to Appendix A for a discussion of forward error correction codes. We summarize here the specific variant that we use in our protocol.

Encoding Hash Packets We will use punctured Reed-Solomon codes to protect the k hash packets for each transmission group from erasures. While it is tempting to philosophically think of a packet as a symbol, the size of each packet, say 1500 bytes, makes for a daunting field size. It is computationally much easier to work in fields of modest size. Accordingly, we partition each hash packet of N bits into N/m symbols (neglecting packet overhead bits and integer round-off for simplicity), each of m bits. Grouping corresponding symbols in each of the k hash packets yields a parallel collection of N/m groups of k source symbols to be fed in parallel to a punctured Reed-Solomon encoder. If erasure protection for losses of up to ℓ authentication packets in a group is desired, we now create ℓ parity packets, with each packet consisting of N/m parity symbols resulting in a parallel collection of N/m groups of $n' = k+\ell$ symbols forming the punctured codewords. Observe that the loss of any packet results in the loss of a symbol in the same location for each of the N/m codewords that have been formed. Thus, as long as no more than ℓ of these n' Hash/Parity (HP) packets are lost, no codewords will have lost more than ℓ symbols and all symbols, hence all hash packets, can then be recovered. See Rizzo [20] for details on this approach.

Encoding Signature Packets Each transmission group creates only one signature packet. Latency considerations prohibit pooling signatures from several groups and as a consequence we end up coding signatures individually from group to group. In such settings repetition codes yield comparable protection to more sophisticated codes and we may as well opt for

simplicity of implementation. A second, more subtle computational factor arises when we consider coding to protect against DoS signature attacks next.

5.2 DoS Protection via FEC

We consider DoS attacks that attempt to confuse the recipient by sending a flood of packets mimicking true packets by usurping their sequence numbers. In what follows, we describe a paradigm that uses FEC to severely diminish such attacks. For instance, using our approach, an adversary launching a DoS signature attack at a rate of 100 Mbps, can be reduced effectively to a 1 Mbps DoS attack with a very small bandwidth overhead. As mentioned earlier, there are two variations of the basic approach. The first one, called *selective sequential verification*, works with repetition codes, and is well-suited for handling signature attacks since they only occupy a tiny fraction of the authentication stream. The second approach, called *selective bin verification*, works with a broad range of FEC schemes, and is particularly well-suited to handle attacks on the HP packets.

Signature Attacks The adversary attack in this instance involves flooding the receiver with spurious signature packets. This is a strong attack since public key verification is the single most computationally expensive step performed at the receiver end. To cope with the attack the sender creates a number of copies, say M, of the first signature packet in a transmission group and interleaves and transmits these copies over the next transmission group. The copies are identical barring the packet sequence number so that the computational burden on the sender is kept minimal.

Selective sequential verification. For a given computational budget, say 5% of the processor, the receiver sequentially samples received packets purporting to be signatures (i.e., with a valid sequence number). Each putative signature is examined with a probability π (determined by the available computational budget and the maximum number of packets arriving per second) so as to ensure that the total number of signature verifications per unit time remains within the budget with high confidence. A sufficient number of replications of the signature vis à vis the adversary's spurious copies will suffice to guarantee a high confidence that a valid signature will, in fact, be discovered. Each time a signature is verified the receiver proceeds to verify the corresponding hashes.

Hash/Parity Attacks An adversary may also attack the HP packets in the authentication stream. Given the relatively large number of HP packets in each transmission group, selective verification, which is based on repetition coding, can no longer be efficiently used because of the large per packet overhead. Selective bin verification provides an efficient way to counter DoS attacks on HP packets.

Selective bin verification. In the absence of a DoS attack, FEC allows us to recover the hash packets for a transmission group as long as any k of the n hash and parity packets in the code successfully arrive at the receiver. Typically, the code parameters are chosen such that the probability of k packets arriving is very high, say, 99.99%. We can instead choose a more powerful code that, with high probability, ensures that at least ck of the HP packets successfully arrive at the receiver (where $c > 1$). Suppose we are searching for k valid HP packets for transmission group i, and suppose we have identified a set W of received packets such that all valid HP packets for transmission group i must be contained within W—this set presumably contains a large amount of adversary traffic. Since we know the valid packet sequence numbers for HP stream for transmission group i, we can easily cluster all of the packets in W by their sequence numbers (*i.e.*, all packets that share a sequence number are put together in a bin) thereby forming several bins. The algorithm now processes these bins in increasing order of sizes—checking all packets in each bin until a valid packet is found. While scanning each bin, we compute the hash of each packet in the bin and match it against the hash included in the signature packet. We stop once we have collected k valid hash and parity packets. Suppose p is the loss probability. If at least ck HP packets arrive and the average size of a bin is W/ck, then searching $[1/(1-p)]k$ bins is likely to yield k valid HP packets. Thus there is a net effect of diminishing the adversary traffic by a factor of $(1-p)c$. So, for instance, choosing c to be 10, can effectively slow down the adversary by a factor of 10, diminishing the computational load on the receiver by an order of magnitude.

5.3 The BAS Protocol

We can now put the previous considerations together to specify the BAS protocol. This protocol uses selective sequential verification to handle a DoS signature attack and selctive bin verification to handle a DoS attack on HP stream. Let N_d, N_h, and N_s respectively denote the number of data packets, HP packets, and signature packets in each transmission group. Also, let $N = N_d + N_h + N_s$, and let R denote the ratio of maximum possible adversary traffic rate to the maximum possible sender traffic rate.

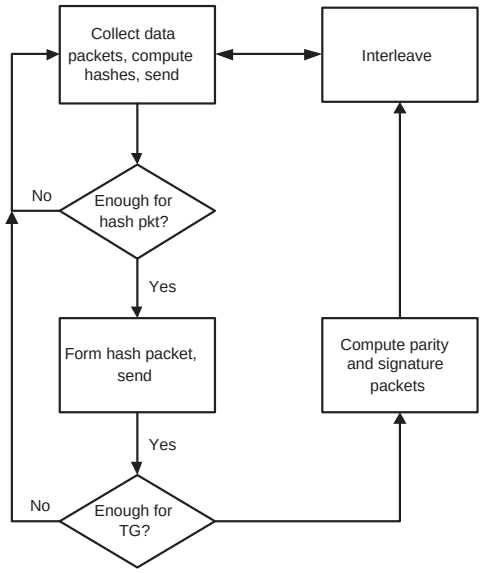

Figure 4: Flow Chart for BAS Sender

Sender Protocol (Figure 4)

1. As data packets are produced, collect their hashes into hash packets. Forward data packets as soon as their hashes are taken, and send hash packets as soon as they are complete.

2. When enough data packets have been processed to make a transmission group, create parity packets and a signature for the group.

3. While sending the next transmission group, interleave the parity packets and N_s copies of the signature packet for this group. In this interleaving, the signature and parity packets should alternate.

Receiver Protocol: Acquisition Phase To *acquire* a stream the receiver begins searching for signature packets as indicated in RTP headers. The acquisition phase is vulnerable to a *replay attack*, and in order to protect against such an attack, the receiver first buffers all candidate signature packets that are received over a window of time that corresponds to the time needed to broadcast g transmission groups. Now it chooses a random subset of packets for verification from this set, picking each candidate packet in the buffer with probability π. Among the signature packets that are successfully verified, the receiver picks the one with the most recent time stamp and proceeds to locate the HP stream corresponding to this packet. At this point, we say that the stream is *acquired.* Once a stream is acquired, the receiver knows the packet numbers that correspond to signature packets sent by the sender since the authentication stream has a repetitive structure.

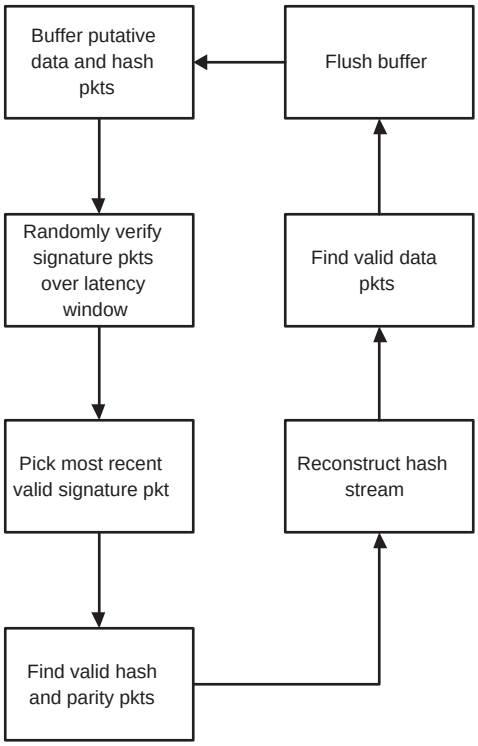

Figure 5: Flow Chart for BAS Receiver

Receiver Protocol: Stream Processing Phase (Figure 5)

1. When a valid signature has been found, it is used to search for the corresponding hash and parity packets in its TG. This search is carried out over a collection of packets received before and after the receipt of the valid signature packet. This collection is called the HP *(HP) verification window* and is chosen to consist of $2(R+1)(N)$ packets received prior to the packet containing a valid signature and $(R+1)N$ packets that are received after that. It is easy to see that all relevant HP packets must be contained within this window.

2. The receiver now applies the selective bin verification technique to cluster these packets into bins (based on packet sequence numbers) and processes these bins in increasing order of sizes. While processing a bin, it checks the hash of each candidate

packet against the hash recovered from the signature packet (using the packet sequence number). Once the valid hash and parity packets in the HP verification window are found, FEC is used to reconstruct as many missing hash packets as possible. Those hashes that can be verified are used to obtain verified data packets, which are passed along or otherwise indicated as authenticated.

3. After a TG has been processed, the receiver continues searching for the next valid signature by verifying each incoming signature with probability π. This process repeats after each valid signature is used to process a TG.

6 Protocol Analysis

A formal analysis of the BAS protocol is presented in Appendix B. We present here a brief overview of how the BAS protocol defends against various attacks. The three attacks of primary interest are signature flooding, replay, and hash flooding.

Signature Flood Signature flood attacks are effectively handled by the selective sequential verification feature of the protocol. For instance, if the sender sends 50 copies of each signature packet, the receiver can sample each candidate signature packet with a 10% probability and still recover a valid signature packet with a 99.5% confidence. On the other hand, the 10% sampling rate cuts down the signature flood by a factor of 10.

Replay Attack A successful replay attack occurs when in the acquistion phase, the receiver chooses a valid old signature packet that is being replayed by the adversary. BAS protects against this by buffering all signature packets seen over a window of several transmission groups. A simple analysis shows that by sampling from a suitably large buffer, we can ensure that the receiver obtains at least one current signature packet with high probability. Since the acquistion phase selects the most recent valid signature packet, the probability of a replay attack is negligible. Once the acquisition phase succeeds in choosing a current signature packet, a replay attack does not effect the stream processing phase. We note here that only a coarse synchronization is needed between the clocks of the sender and the receiver in order to choose the most recent signature packet.

Hash Flood If the adversary attack rate is so high that we cannot perform even hash computations of all incoming adversary and valid packets, then we essentially drop data packets (the ones that could not be authenticated). It is not difficult to see that an adversary targeting hash computation load at the receiver will be more effective by focusing on the hash parity stream than data stream. Each valid data packet for which receiver is unable to compute a hash due to the computational overload, causes a loss of a single data packet. In contrast, each transmission group for which we are unable to compute hashes for enough packets in the hash parity stream, will cause us to lose all N_d data packets in the corresponding transmission group.

Other Attacks Other attacks of interest include: integrity attacks on the data or HP stream, attacks on key distribution, and attacks on time stamps. Let us first consider each of these in turn. If an adversary introduces false (non-replay) packets into the data or HP stream, then their hashes will not coincide with the ones signed by the signature packets. An adversary who sends a large number of such false packets will have them discarded as quickly as their hashes can be computed. The BAS protocol uses public keys so there is no session key distribution protocol to attack. Attacks on PKI represent the same risk for other protocols that they do for BAS. BAS uses a very coarse time requirement, on the order of hours, so attacks on time stamps cannot exploit fine errors.

7 Implementation

We have implemented the BAS protocol based on the Real-time Transport Protocol [23]. RTP provides a 12 byte header that includes a 16 bit sequence number, a timestamp, and other information such as a synchronization source identifier. The BAS protocol exploits the sequence numbers to correlate three RTP streams. The first of these, the data stream, is an RTP stream of any kind provided by an application-level protocol. BAS associates two additional RTP streams to this one, namely the HP and signature streams. A given collection of data packets will determine corresponding hash, parity, and signature packets; this associated collection of data, hash, parity and signature packets is a Transmission Group (TG). The nature of a transmission group depends on the rate of at which the data will be sent and the anticipated reliability of its channel. We developed design parameters for two rates, 10Mbps and 100Mbps, and three levels of reliability reflecting average losses of 5%, 20% and 40%. The parameters are predicated on the assumptions that the receiver should be able to verify the packets it receives within 2 seconds over a low speed channel (10Mbps)

and 1 second over a high speed channel (100Mbps) with a confidence of 99%. For a given rate and reliability, these assumptions yield a choice of the number of data packets k in a transmission group, the number of hash packets needed to hold their hashes, and a number of parity packets ℓ needed to provide sufficient reliability to the hash stream. Table 3 provides reliability parameters used in our implementation. On a 10Mbps link we use 11 hash packets: for loss rates of 5%, 20%, and 40% we use 5, 11, and 22 parity packets respectively. On a 100Mbps link we use 57 hash packets: for loss rates of 5%, 20%, and 40% we use 10, 30, and 66 parity packets respectively. The size of the transmission group is the number of data packets assuming 144 for each hash packet plus the number of hash and parity packets plus the number of signature packets. For 10Mbps we send 20 signature packets and for 100Mbps we send 200. A TG for 10/5 (10Mbps and 5% loss) has 1620 packets, including 20 signature packets, while a TG for 100/5 has 8475 packets, including 200 signature packets. These choices assume that packets can contain up to 1500 bytes of data. Nothing about the BAS algorithm fundamentally relies on packets of 1500 bytes, but we would use different parameters for other sizes. For a hash we use the first 10 (of 20) bytes in a SHA hash [8].

The number of signature packets to be sent is determined by the need to ensure that the receiver can find a valid signature packet within the specified verification latency assuming that an attacker is given the same capacity as the sender. Signatures use 1024 bit RSA with an exponent of 17. Selective sequential verification uses a verification frequency parameter of 25% for 10Mbps and 2.5% for 100Mbps.

BAS relies on the use of sequence numbers to determine transmission groups from the three streams of packets. We assume that all packets are given IP, UDP, and RTP headers as illustrated in Figure 6(a). The sizes of headers and available payload are given in bytes in the figure. Hash packets are illustrated in Figure 6(b). In this and other cases in the figure, HLH refers to the IP, UDP, and RTP headers. Following recommendations in the RTP standard, the HL and signature streams use distinct UDP ports to distinguish them from each other and the data stream. In particular, the three streams have distinct sequence numbers. The SN field of each hash packet is the first sequence number from the data stream for which a hash appears in the hash packet payload, and the range indicates the total number of data packets whose hashes are included. This number is generally 144 (the maximum that will fit), but could be smaller. The hash of a data packet applies to its RTP header and payload but not to its IP and UDP headers. Parity packets are illustrated in Figure 6(c). Parity packets are created from applying a systematic Reed Solomon code to the RTP and payload portions of the hash packets from their transmission group. These packets have their own RTP headers, but the parity information does not recover these or the IP and UDP headers of hash packets in the transmission group. It also does not allow the recovery of any lost data packets. Signature packets are illustrated in Figure 6(d). The signature is applied to the truncated hash of the significant fields in the packet following the signature. These include the sequence number field, SN, which is the RTP sequence number of the first hash packet in its group, the HRng field, which is the value of k, the PRng field, which is the value of l, a 32 bit time based on the Network Time Protocol (NTP), and the collection of truncated hashes of the HL stream packets in the transmission group. These hashes cover all of the data in these packets except for the IP and UDP headers. Note that only one signature is required for each transmission group, no matter how many signature packets are used.

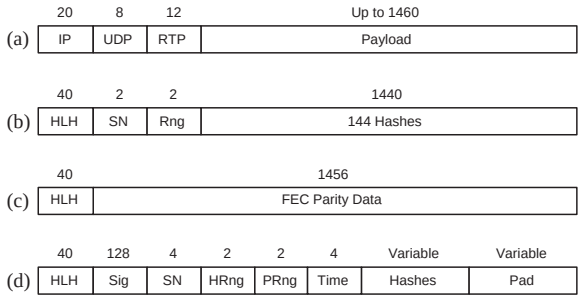

Figure 6: Packet Formats

The time stamp is used to avoid replay attacks based on old sequence numbers. With 16 bit sequence numbers there may be some wrap-around in the signature stream, but this will very slow, on the order of many hours. The receiver checks that the time stamp is accurate to within 20 minutes to prevent replays. We could have used the time stamp in the RTP header, but this complicates interactions with RTP since the implementation uses the same signature for all of the signature packets in a transmission group whereas RTP will assign them all distinct time stamps.

8 Experiments

We now present an experimental evaluation of the BAS protocol for various settings of rate, loss and attack.

8.1 Setup

We assume the existence of a DoS attacker with access to the various levels of bandwidth. The interesting independent variables are the following: sender rate and latency; loss rate; average burst length; attacker rate. Our approach fixes a target latency; the protocol is then designed for various bandwidth and reliability characteristics of the channel. The interesting dependent variables are: sender throughput; receiver throughput; bandwidth overhead; authentication loss. The sender rate is the number of megabits of data packets that can be processed in one second; processing consists of producing the necessary hash, parity, and signature packets for the data packets. The receiver rate is the rate in megabits per second at with which valid data packets can be recovered from a mixture of packets originating from the sender and an attacker. The bandwidth overhead is the percentage of bandwidth devoted to hash, parity, and signature packets. The authentication loss is the percentage of data packets received by the receiver that cannot be verified by the receiver due to the loss or reordering of hash, signature, and parity packets.

To carry out an experiment for a sender we start with a collection of data and simply generate the necessary hash, signature, and parity packets. For a receiver, we start with the stream produced by a valid sender and apply a loss model to remove a subset of the stream. We then insert DoS packets randomly into the resulting stream at a ratio determined by the attack factor. DoS packets are assumed to take advantage of an informed attack, so, for instance, they assign sequence numbers that the receiver is expecting to see from the sender. Our experiments are for DoS attacks *based on signature flooding only*. In particular, no other kinds of packets are sent by an adversary. The aim is to measure robustness against signature floods even at levels where the adversary could be effective by attacking another limit. For instance, a receiver can perform hashes on about 77,000 packets each second so a factor 10 attack on a 100Mbps link would overwhelm this capacity if it forced the receiver to perform hashes on all of the packets it receives.

Our throughput numbers are based on the average of three runs for 32 transmission groups. The authentication loss numbers are based on 15 runs over 32 transmission groups. These were done on a 2.4GHz PC with enough memory to avoid using the disk and Redhat Linux 7.3 using cryptographic operations from OpenSSL 0.9.6.

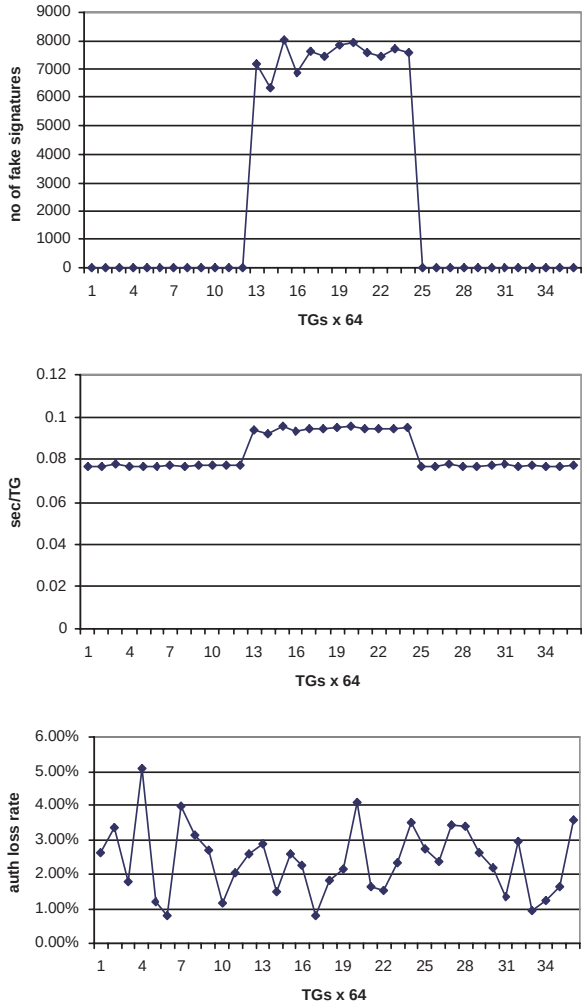

Figure 7: Scenario Attack Profile

8.2 Results

We begin with a collection of figures derived from a scenario that illustrates the basic measurements we considered. Assume we are given a sender that sends packets at a rate of 10Mbps and a receiver that gets them over a channel with a 20% average loss that occurs in bursts that average 100 packets. After a period of time an attacker sends fake signature packets which are received at a rate of 8Mbps. For this profile the BAS implementation uses transmission groups of 1626 packets, including 20 signature packets. Each TG takes about 2 seconds to transmit. Signature packets are checked at the receiver with a probability of 25%. Figure 7 illustrates a few of the associated effects of this attack. Figure 7(a) shows the number of false signa-

Table 2: Disproportionate DoS Signature Attack Parameters for 20% Loss

Sender Rate (Mbps)	10	100	100
Attack Factor	10	5	5
PKC Budget Per TG	800	400	1000
Packets Per TG	1666	8333	8333
Signature Copies	131	657	249
Checking Probability(%)	4	1	2
Expected Sender Packets(%)	7	14	14
Signature Overhead(%)	8	8	3

tures checked per collection of 64 transmission groups. Figure 7(b) shows the time required to process each collection of 64 transmission groups. Note that this cost rises slightly during the attack. Figure 7(c) shows the authentication loss rate, that is, the percentage of data packets that reached the sender but could not be authenticated because of the loss of packets in the authentication stream.

We now provide a sample of the figures we derived from various additional experiments. Figure 8 shows processing throughputs for independent loss rates with no DoS attack. These experiments use the parameters in Table 3. By contrast Figure 9 shows throughputs for independent loss for a collection of different disproportionate attacks using the signature repetition codes prescribed by the theory in Section 6. The first three pairs of experiments are for a factor 10 attack on a 10Mbps sender at various loss rates assuming a budget of 800 Public Key Checks (PKC) per TG. The next three pairs are for a factor 5 attack on a 100Mbps sender with a 400 PKC/sec budget. The last three pairs are for a factor 5 attack on a 100Mbps sender with a 1000 PKC/sec budget. Table 2 shows the corresponding parameters for the case of 20% loss as derived from our analysis. These figures are based on a 99% confidence for authentication.

Figure 10 shows processing throughputs for correlated losses ranging from an average of 10 packets per burst loss to 200 packets per burst loss using a two-state Markov chain. As in Figure 8 we are assuming a proportionate attack. Sender rates are not affected by burst rates so they remain the same as for the independent case. Figure 11 shows authentication loss rates plotted over burst rates. For example, if a 10Mbps channel is used at 5% loss, then authentication losses begin to occur at bursts of about 80.

8.3 Analysis

The scenario analyzed in Figure 7 shows that an attack causes the receiver to check an increased number of bad signature packets, but this number is far less than the number actually sent since packets are checked with a probability of 25% and signature packets for a given TG can be ignored after a valid one is found. The time required to prcess a TG rises during an attack, but the added processing time is only about .02 second per TG. This is about 1% of the processor time, which is close to what would be predicted by Figure 7(a) and Table 1. Figure 7(c) shows that the authentication loss is between 2% and 3% and is not much affected by the attack.

When there is no DoS attack, receiver rates are slightly better than sender rates. A receiver does not need to process parity packets unless some hash packets are lost, so the performance of a receiver is better when there is a low loss rate. A sender needs to create more parity packets when the channel is lossy, so higher reliability leads to better performance for both the sender and receiver. A higher rate allows more amortization of signature costs and different FEC characteristics, so processing throughput rates are better for higher bandwidths. These observations address the slight decrease in thoughputs in Figure 8 for lower bandwidths and reliability.

When a DoS attack occurs, receiver throughputs are significantly degraded as a function of the factor of the attack. Figure 9 illustrates how the choice of parameters can take advantage of a tradeoff between bandwidth overhead and throughput. For instance, the 1000 PKC/sec limit for the 100/20 profile yields a lower throughput at the receiver than the corresponding 400 PKC/sec limit but reduces the signature overhead from 8% (657 signatures per TG) to 3% (249 signatures per TG) as shown in Table 2. The decline in throughput is so small that it seems reasonable to use larger processing budgets to reduce bandwidth overhead.

Our reliability parameters were based on independent loss, but their throughput is not much affected by correlated losses, as is shown by Figure 10. When there is a DoS attack the throughputs become lower and more similar for different reliability levels. Under attack a greater percentage of processor effort will be devoted to checking false signatures so differences in FEC costs will matter less.

As shown in Figure 11, athentication losses are nearly absent for bursts below about 100 for all of the profiles, but rise significantly between 100 and 180. At bursts of 200 or more authentication begins to break down significantly. We did not include a graph to show this, but adding more parity packets decreases authen-

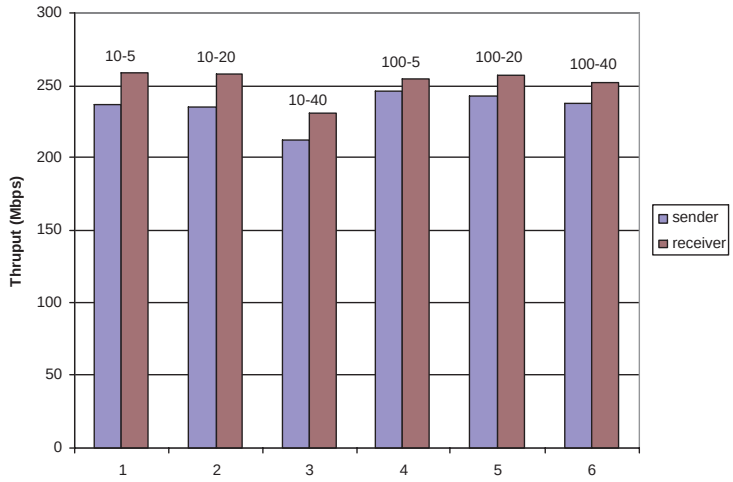

Figure 8: Throughputs for Independent Loss Under No DoS Attack

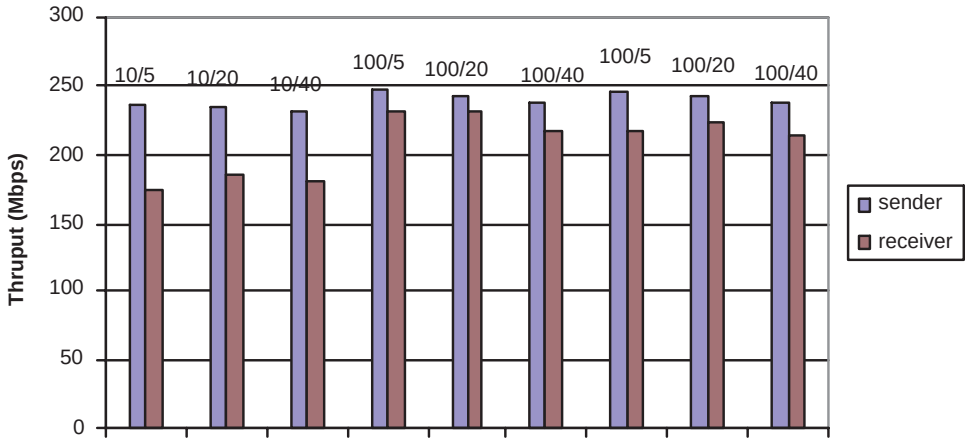

Figure 9: Throughputs for Independent Loss Under Disproportionate DoS Signature Attacks

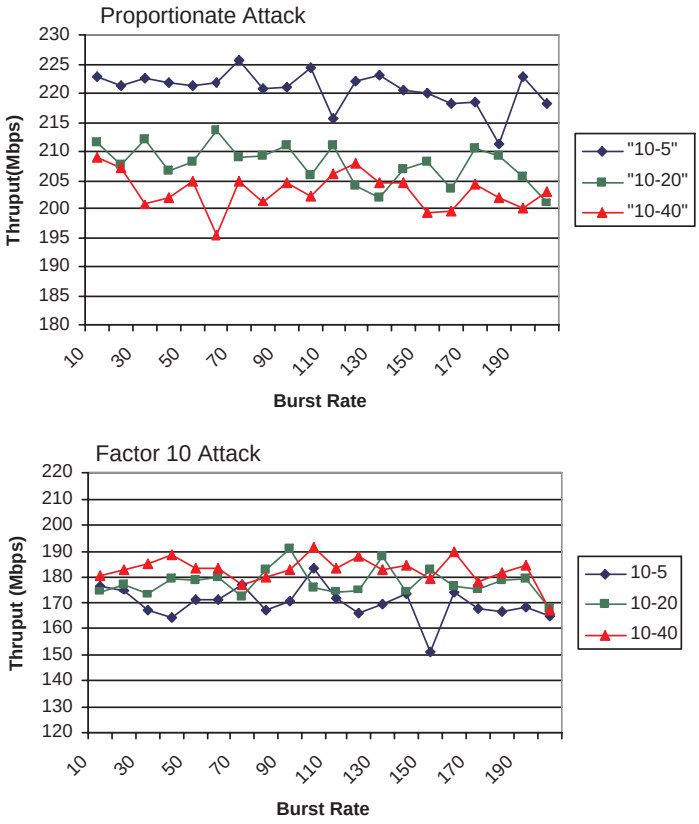

Figure 10: Receiver Throughputs for Correlated Loss Under DoS Signature Attacks

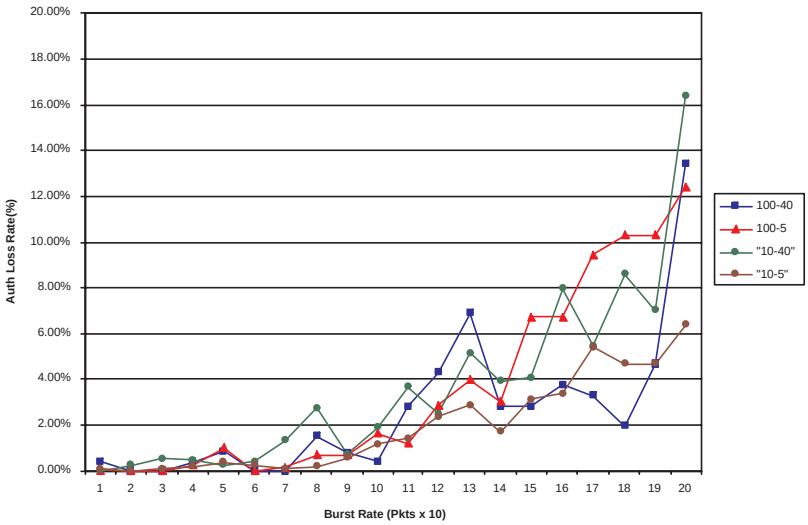

Figure 11: Authentication Loss over Correlated Loss Under Proportionate DoS Signature Attack

tication losses significantly at higher burst levels. In particular, authentication losses are mainly due to the loss of parity packets rather than signature packets.

9 Conclusion

Broadcast authentication streams provide simplicity, efficiency, and robust protection against denial of service at the cost of modest additional latency and authentication loss. BAS requires that the rate of the sender is known within general bounds and upper bounds can be placed on the loss rate and the bandwidth available to an adversary. It requires only familiar and well-understood FEC and cryptographic operations such as Reed-Solomon codes, SHA hashes, and RSA signatures. BAS has bandwidth overhead and performance as good as any previous approach to public key authentication of broadcast streams and gives DoS protection that is significantly better than any previous approach. The DoS protection in BAS is based on a new technique called selective verification which uses redundancy and probabilistic verification to diminish the adversary attack rate by large factors. We have developed theoretical foundations for selective verification and shown its effectiveness in experiments over a broad range of communication characteristics.

Acknowledgements

We appreciated assistance from programmers who helped us with coding the BAS protocol and collecting data for the experiments in Section 8. Sumeet Bedi led this effort with assistance from Watee Arjsamat, Sonal Ghandi, Kevin Lux, and Aloka Singh. We also received useful comments from Karthikeyan Bhargavan, Ran Canetti, Michael Greenwald, Klara Nahrstedt, Adrian Perrig, and Jonathan Smith. We thank Luigi Rizzo for assistance and the use of his FEC package. Research of Gunter and Tan was supported in part by NSF EIA00-88028 and ONR N00014-02-1-0715. Research of Khanna was supported in part by an Alfred P. Sloan Research Fellowship and by an NSF Career Award CCR-0093117

References

[1] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, 1999.

[2] H. Chernoff. A measure of the asymptotic efficiency of tests of a hypothesis based on a sum of observations. *Ann. Math. Stat.*, 23:493–507, 1952.

[3] Danny Dolev and Andrew C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.

[4] David C. Feldmeier, Anthony J. McAuley, Jonathan M. Smith, Deborah S. Bakin, William S. Marcus, and Thomas M. Raleigh. Protocol boosters. *IEEE Journal on Selected Areas in Communications*, 16(3):437–443, 1998.

[5] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Proceedings of Crypto'97*, pages 180–197, 1997.

[6] Philippe Golle and Nagendra Modadugu. Authenticating streamed data in the presence of random packet loss. In *Proceedings of the NDSS Symposium*, 2001.

[7] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc.*, 58:13–30, 1963.

[8] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. RFC 2104, IETF, February 1997.

[9] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. Forward error correction (fec) building block. RFC 3452, IETF, December 2002.

[10] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[11] R. J. McEliece. *The Theory of Information and Coding*. Addison-Wesley, 1977.

[12] Ralph Merkle. A certified digital signature. In *Proceedings of Crypto'89*, pages 218–238, 1990.

[13] Sara Miner and Jessica Staddon. Graph-based authentication of digital streams. In *IEEE Symposium on Security and Privacy*, pages 277–288, 2001.

[14] A. Pannetrat and R. Molva. Efficient multicast packet authentication. In *Proceedings of the NDSS Symposium*, 2003.

[15] J. M. Park, E. K. P. Chong, and H. J. Siegel. Efficient multicast stream authentication using erasure codes. *ACM Transactions on Information and System Security*, 6(2):258–285, 2003.

[16] Adrian Perrig. The biba one-time signature and broadcast authentication protocol. In *ACM Conference on Computer and Communications Security*, pages 28–37, 2001.

[17] Adrian Perrig, Ran Canetti, Dawn Xiaodong Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the NDSS Symposium*, 2001.

[18] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[19] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer, 2003.

[20] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, 1997.

[21] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *ACM Conference on Computer and Communications Security*, pages 93–100, 1999.

[22] S. M. Ross. *Stochastic Processes*. Wiley, second edition, 1996.

[23] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, IETF, January 1996.

[24] Wong and Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7, 1999.

A Forward Error Correction

Several unicast protocols utilize a combination of a low complexity error-detection code (typically a cyclic redundancy check (CRC) code) coupled with feedback in the form of automatic repeat requests (ARQs) to recover from errors and losses in transmission. Such ARQ-based protocols, while well understood, unfortunately do not scale well in multicast settings. In such cases, coding introduced in the transmission to cope with errors and losses in the authentication stream must be sufficiently powerful to allow each recipient to unambiguously reconstruct the entire authentication stream corresponding to a group of data with very high confidence (so that each recipient loses only a very small fraction of transmissions to non-authentication). As feedback is eschewed, the entire process of coding, transmission, and decoding is accomplished in a single forward pass.

In general, error-control coding adds redundancy to the stream. Repetition codes, for instance, constitute a naïve application of this idea whereby each packet to be protected is simply retransmitted a fixed number of times. From the perspective of erasure recovery, however, repetition codes have simplicity commending them but little else. As we see in the next section, it is easy to get a five- to ten-fold improvement in coding overhead over repetition codes with a modicum of effort.

For our present purposes we focus on a very powerful and practical family of codes called the *Reed-Solomon (RS) codes* over the finite field $GF(2^m)$ (c.f., MacWilliams and Sloane [10] or McEliece [11], for instance) for some positive integer m. An $RS(n, k)$ code will consist of $n = 2^m - 1$ symbols in $GF(2^m)$ out of which the first k are source symbols and the remaining are parity check symbols that add redundancy.

Several features of the Reed-Solomon codes make them particularly appealing for our application. The codes can be implemented very efficiently using public domain software. No special tweaking or proprietary material is needed. The codes are very well suited to situations where errors in bit transmissions occur in bursts. And the Reed-Solomon codes can be efficiently combined or *concatenated* with other codes to form even more powerful codes.

Most importantly, an $RS(n, k)$ code can recover from *any* combination of up to $n - k$ erasures. Suppose it is desired to provide erasure protection for up to ℓ erasures out of k source symbols. Then any Reed-Solomon code with length $n \geq k + \ell$ and dimension k will provide the requisite level of erasure protection and more. Indeed, if $\ell \ll n - k$, the code protects against much more than the requisite ℓ erasures. In this case one can elect to keep just, say, the first ℓ parity check symbols in each codeword together with the k source symbols and cheerfully drop the remaining $n-k-\ell$ parity check symbols. The resulting code is called a *punctured* Reed-Solomon code; in the punctured code, each codeword consists of only $k + \ell$ symbols. We can still afford to lose up to ℓ more symbols before the code falters. The benefit of puncturing the code thus is that we have now boosted the rate of the code to $k/(k + \ell)$ which may effect a substantial reduction in coding overhead over the original k/n rate. Thus, erasure protection may be continuously traded-off against rate by puncturing the code.

Other codes with roughly similar characteristics such as Rabin's Information Dispersal Algorithm may also be used. While coding overheads in our application are typically low and coding and decoding costs are not significant, in situations where decoding cost is important one may trade-off space for time using codes such as Tornado codes. For these and related ideas see Luby, et al [9], Park, et al [15], and Rizzo [20].

B Formal Analysis of BAS

In this section we describe the theoretical foundations of the protocol and provide analyses of different loss and DoS attack models.

B.1 Loss Recovery Analysis

The simplest loss model occurs when packets are assumed to be dropped independently with some fixed probability p. Suppose the hash/parity stream of a transmission group consists of n' packets comprised of k hash packets and $\ell = n' - k$ parity packets. As before, we suppose that the ℓ parity packets are obtained by puncturing an (n, k) systematic Reed-Solomon code and selecting $\ell \leq n - k$ parity packets. Consider the transmission of the $n' = k+\ell$ hash/parity packets over a packet erasure channel with packet loss probability p. Assume for the nonce that there are no packet insertions, i.e., no DoS attack on the hash/parity stream.

Let the integer-valued random variable S denote the number of dropped packets in the hash/parity stream of the transmission group. We can interpret the probability that S does not exceed ℓ as our *confidence* in the recoverability of all hash packets corresponding to a given transmission group. Alternatively, the confidence represents the long-run fraction of transmission groups that are authenticatable, DoS attacks in abeyance, in the sense that the hash packets of these groups are recovered. We typically require a confidence of at least

	$k=11$			$k=57$		
p	ℓ	$(\ell_{\rm H})$	ℓ'	ℓ	$(\ell_{\rm H})$	ℓ'
0.05	5	(8)	22	10	(17)	114
0.20	11	(12)	44	30	(33)	285
0.40	22	(22)	77	66	(67)	513

Table 3: Coding overhead for Reed-Solomon and repetition codes.

99 percent that all hash packets in a group are recovered so that fewer than one in a hundred groups are compromised.

As packet drops are independent, S conforms to a binomial distribution and a selection of exponential tail bounds may be deployed to provide crisp estimates for the tail probability,

$$\Pr\{S > \ell\} < \exp\Big\{-(k+\ell)D\big(\tfrac{\ell}{k+\ell} \,\big\|\, p\big)\Big\}$$

(Chernoff bound)

$$< \exp\left\{\frac{-2\big((1-p)\ell - kp\big)^2}{k+\ell}\right\},$$

(Hoeffding bound)

where, with logarithms to base e,

$$D\big(\tfrac{\ell}{k+\ell} \,\big\|\, p\big) = \tfrac{\ell}{k+\ell}\log\tfrac{\ell}{p(k+\ell)} + \tfrac{k}{k+\ell}\log\tfrac{k}{(1-p)(k+\ell)}$$

denotes the Kullback-Leibler divergence between the probability distributions $\big(\frac{\ell}{k+\ell}, \frac{k}{k+\ell}\big)$ and $(p, 1-p)$. The first bound is that of Chernoff [2], the second, slightly more analytically amenable, is due to Hoeffding [7]. Given a value of k and a desired confidence $1-\delta$ (where $\delta = 0.01$, say) one typically wishes to determine a value of ℓ for which the desired confidence is attained. Inverting Hoeffding's bound, for instance, we obtain that *if the number of parity packets per transmission group satisfies*

$$\ell \geq \frac{kp}{1-p} - \frac{\log\delta}{4(1-p)^2}\left[1 + \sqrt{1 - \frac{8k(1-p)}{\log\delta}}\,\right] \qquad (1)$$

then all k hash packets in a given transmission group can be recovered with confidence at least $1-\delta$. As a practical matter, the values of ℓ provided by Hoeffding's bound are only slightly larger than those provided by Chernoff's bound; see Table 3 where, for a given value of k, ℓ denotes the Chernoff estimate of the parity overhead, $\ell_{\rm H}$ denotes the Hoeffding estimate, and for comparative purposes, ℓ' is the parity overhead that results if a naïve repetition code were to be used in lieu of a Reed-Solomon code.

More sophisticated Markovian loss models can be constructed to capture time correlations in packet losses. Similar results can be derived for these models though we will not describe them here in the interests of brevity.

B.2 Denial of Service Attacks

A DoS attack may focus on the data stream, hash/parity stream, or signature stream. Or an attacker may elect to spread his resources in an attack on some combination of these streams. In a shared channel model the attack is in the form of a flood of spurious packets having the apparent characteristic of the packets in the stream under attack, for instance, by bearing sequence numbers of legitimately expected packets.

We may consider attacks on each of the three streams in isolation; a combination attack effectively reduces the attack factor of the adversary in each stream as his resources have to be spread across the streams. A successful attack on a data packet will result in the effective loss of that packet as it cannot be verified; a successful attack on a hash packet will result in the invalidation of the group of data packets whose hashes have been compromised; a successful attack on a signature packet will compromise the entire transmission group. Of such attacks, a signature flood attack is potentially the most damaging as an adversary can invalidate an entire transmission group in one fell swoop if the attack is successful. We hence begin with an analysis of signature flooding attacks and how the BAS protocol copes with such attacks.

As a design parameter we require that the protocol provide a guaranteed confidence of $1-\delta$ that any given transmission group is verifiable. Here $\delta \in (0,1)$ is our *confidence parameter*. We may think of δ as some suitably small number (we selected $\delta = 0.01$ in our simulations) but for purposes of analysis we leave this as an application-specified parameter. Two additional design parameters are the authentication and loss recovery *overhead* o which is the fraction of packets in a transmission group devoted to authentication and error or loss recovery and the sender-side authentication *latency* τ which is the delay between the beginning of transmission of data in a transmission group and the completion of transmission of packets needed to authenticate the group. Of course, the overhead and latency are related. In our analysis we assume that the maximum overhead o is specified as a design parameter; the latency τ is then determined as a function of o and δ. As will become clear, we could equally well have specified a maximum latency that the application can

withstand and determine the overhead that is incurred in consequence. The latter may be more appropriate in situations where excess capacity is available.

For definiteness we consider the following explicit specifications of communication and computation parameters in the analysis. Suppose all packets have a fixed length and consider a continuous transmission setting where a sender has an available bandwidth of W packets per second which he fully utilizes. We consider a shared channel model where the attacker has the capability of transmitting at the rate of RW packets per second where R is the dimensionless *attack factor*. We assume that sender packets arrive in order at the receiver (though adversarial packets may be timed and inserted anywhere in the sequence) and that the receiver has a computational budget of $K = K_s$ signature checks per second on average. Finally, we suppose that packet transmissions are over an independent loss channel in which each packet transmitted by the sender is lost independently with probability p. We do not assume that the adversary's packets are also subject to loss. Typical numbers that we consider in the examples and simulations are 1500 byte packets; transmissions over 1 Mbps, 10 Mbps, and 100 Mbps connections, which translate into packet transmission rates of $W = 83$, 833, and 8333 packets per second, respectively; proportionate and disproportionate attacks with attack factors $R = 1$ and $R = 10$, respectively; computational budgets K between 40 to 800 public key checks per second on stock PC's assuming that we can devote 5% to 10% of the processor toward signature checks; packet loss probabilities p of 5%, 20%, and 40%; confidence parameter $\delta = 0.01$; and overhead o between 5% and 10%.

The analysis may be modified to fit other models such as variable sender rates, attacks on shared/modification channels, out-of-order packets, and correlated packet losses without much ado.

In this setting our goal is to determine the size and composition of a transmission group that meets the confidence and overhead constraints. Let N (to be determined) denote the number of packets in a transmission group and let N_d, N_h, and N_s denote the number of data packets, hash and parity packets, and signature packets, respectively, comprising the transmission group. Once determined, the size the transmission group specifies the sender-side authentication latency, $\tau = N/W$. (Of course, this is for the independent loss model; in a correlated loss model, the authentication packets will need to be interspersed across the following transmission group to combat burst errors leading to a latency of approximately $2N/W$.)

Signature Stream

The number of signature copies N_s needed per transmission group will be determined loosely by the loss rate and the attack factor. The actual estimates depend on whether the sequential verification or bin verification protocols are adopted and we consider these in turn.

Selective sequential verification Suppose that each incoming signature is verified independently with probability π. The following result is elementary but useful.

Lemma 1 *The receiver will successfully verify a signature in any given transmission group with confidence at least* $1 - \delta$ *provided* $N_s \geq \log(\delta)/\log\big(1 - (1-p)\pi\big)$. A fortiori *it suffices if* $N_s \geq -\log(\delta)/(1-p)\pi$.

Proof: Consider the start of the transmission of the current group of signature packets (i.e., N_s copies of the same signature packet). The probability that a given signature packet is both successfully received and verified by the receiver is given by $(1-p)\pi$. It follows that the probability that none of the *bona fide* signature packets in the current group is successfully verified is $\big(1-(1-p)\pi\big)^{N_s}$ and we require this to be no larger than δ. The bound on N_s follows. Finish off the proof with the elementary observation $-\log(1-x) > x$ for $0 < x < 1$. □

As the entire transmission group requires N/W seconds for transmission, under an informed DoS attack on the signature stream with attack factor R, the maximum number of packets in the signature stream of the current group is $N_s + RN = R(N_d + N_h) + (R+1)N_s$. To keep within the computational budget of K signature checks per second on average at the receiver, it suffices hence to verify signatures randomly with probability

$$\pi = \frac{KN/W}{N_s + RN} = \frac{K(N_d + N_h + N_s)}{WR(N_d + N_h) + W(R+1)N_s}. \quad (2)$$

(More accurately, π is chosen as the smaller of the above quantity and 1; to obviate trivialities suppose that the right-hand side above is less than 1.) A direct application of the lemma shows that a choice of $N_s \geq -\log(\delta)/(1-p)\pi$ will provide adequate protection against a signature flood DoS attack while staying within the given computational resources. It follows that the number of signatures N_s may be chosen to be

any positive integral value for which the quadratic

$$Q(N_s) = (1-p)KN_s^2 + \big[(1-p)K(N_d+N_h) - W(R+1)\log\tfrac{1}{\delta}\big]N_s - WR(N_d+N_h)\log\tfrac{1}{\delta} \stackrel{\text{def}}{=} aN_s^2 + bN_s + c$$

is nonnegative. It is easy to verify that Q is convex and has two real roots. Indeed, the discriminant of the quadratic is nonnegative and given by

$$\Delta = b^2 - 4ac \le \big[(1-p)K(N_d+N_h) + W(R+1)\log\tfrac{1}{\delta}\big]^2.$$

As we may choose N_s to be any integral value greater than or equal to the larger of the roots of Q, it suffices if

$$N_s \ge \frac{-b+\sqrt{b^2-4ac}}{2a} = \frac{-b+\sqrt{\Delta}}{2a}$$

and by virtue of the above bound on the discriminant we hence obtain the following

Theorem 2 *Under a signature flood attack with attack factor R in a shared channel with independent packet loss probability p, selective sequential verification at a rate of K signature checks per second on average will result in the acquisition of a valid signature for any transmission group with probability at least $1-\delta$ if*

$$N_s \ge \frac{W(R+1)\log\frac{1}{\delta}}{(1-p)K}.$$

Observe that, as anticipated, the number of signature copies in a transmission group is determined by the attack factor R and the loss rate p and that, moreover, N_s may be specified independently of N_d and N_h.

Selective bin verification The probability that none of B selected bins contains a valid signature is p^B. Thus, we can ensure that the probability of encountering a valid signature in the group of bins corresponding to a transmission group is at least $1-\delta$ provided $B \ge \log\delta/\log p$. Consider a signature flood attack with attack factor R over a shared channel. Condition on the number of consecutive transmission groups for which verification failed following the last successfully validated transmission group. The window of signature packets given $j \ge 0$ consecutive group verification failures is then bounded above by $(j+2)RN+N_s$ packets. Selective bin verification proceeds by examining the B smallest bins in order of size. The total number of signature packets in these B bins is then bounded by $B\big[(j+2)RN+N_s\big]/N_s$. As the probability of j consecutive verification failures is bounded above by δ^j, the expected number of signature checks for the current group is bounded above by

$$B\sum_{j=0}^{\infty}\left(\frac{(j+2)RN+N_s}{N_s}\right)\delta^j = \frac{B\big[R(N_d+N_h)(2-\delta)+\big(R(2-\delta)+1-\delta\big)N_s\big]}{N_s(1-\delta)^2}.$$

It suffices if the right-hand side is bounded above by KN/W to keep within the computational budget. This leads to a quadratic inequality for N_s and proceeding as in the proof of Theorem 2, we obtain

Theorem 3 *Under a signature flood attack with attack factor R in a shared channel with independent packet loss probability p, selective bin verification over $B = \lceil \log\delta/\log p\rceil$ bins will result in the acquisition of a valid signature for any transmission group with probability at least $1-\delta$ if*

$$N_s \ge \frac{W\big[R(2-\delta)+1-\delta\big]\log\frac{1}{\delta}}{K(1-\delta)^2\log\frac{1}{p}}.$$

The mean rate of signature checks will be bounded by K.

Observe that the signature overhead imposed by the two strategies is very comparable.

Hash/Parity Stream

The number of hash packets k in a transmission group is proportional to the number of data packets in the group, $k = \alpha N_d$ for some constant α. If, for instance, packets have size 1500 bytes and for hashes we use the first 10 bytes (of 20) in a SHA hash, then each hash packet of 1500 bytes contains the hashes of 144 data packets (some packet real estate to the tune of 56 bytes being consumed by various headers) and thus, $k = N_d/144$. The number of parity packets ℓ requisite, however, in the absence of a DoS attack depends on the channel loss probability as seen in the estimate (1) though, as seen, ℓ only grows linearly with k, hence with N_d. In the absence of a hash/parity flood attack, the total number of packets $N_h = k+\ell$ required in the hash/parity stream increases linearly with the number of data packets N_d. The typical overhead is less than a couple of percent as we see in Table 3.

Hash/parity flood attacks may be handled on one of two levels. Hash computations are typically an order of magnitude faster than signature verifications; a receiver can compute of the order of $77,000$ packet hashes per second but only about 8000 signature checks per

second on a stock PC. If the attack factor R and available bandwidth W are both moderate (for instance, a proportional attack on a 10 Mbps link), the receiver can simply compute the hash of all packets in the hash/parity stream discarding spurious packets (assuming that a valid signature has been acquired). If the attack factor R exceeds the capacity of the receiver K_h to perform all the hashes, a variation on the bin strategy allows us to force the adversary to diffuse his advantage at small cost in overhead. The idea is to select a number of parity packets large enough that the the lowest population B bins contain at least k valid packets from the hash/parity stream. As packet losses are independent, Hoeffding's bound quickly allows us to estimate the requisite number of bins,

$$B \geq \frac{2(1-p)k + \frac{1}{2}\log\frac{1}{\delta} + \sqrt{2(1-p)k\log\frac{1}{\delta} + \frac{1}{4}\log^2\frac{1}{\delta}}}{2(1-p)^2},$$

to ensure that at least k of the B selected bins contain valid hash/parity packets.

The analysis now parallels that for the number of signatures in selective bin verification and we obtain the following

Theorem 4 *Selective bin verification yields at least k packets in the hash/parity stream with confidence at least $1-\delta$ if the number of packets in the hash/parity stream satisfies*

$$N_h = k + \ell \geq \frac{BW\left[R(2-\delta) + 1 - \delta\right]\log\frac{1}{\delta}}{K_h(1-\delta)^2\log\frac{1}{p}}. \qquad (3)$$

The number of parity packets ℓ that are requisite can now be determined as the larger of the values determined from (1) and (3).

Data Stream

The number of packets in the data stream of a transmission group can now be determined. Recall that a design parameter is the maximum allowable transmission group authentication overhead $o \geq (N_h+N_s)/N = 1-N_d/N$ which is typically specified by the application. The number of packets N_d in the data stream hence is required to satisfy $N_d \geq (1-o)N = (1-o)(N_d+N_h+N_s)$. In line with the discussion for the hash/parity stream, we may set $N_h = cN_d$ for a constant c so that we obtain the following

Theorem 5 *A choice of*

$$N_d \geq \frac{(1-o)N_s}{1-(1-o)(1+c)}$$

data packets in a transmission group together with $N_h = cN_d$ hash/parity packets guarantees that the overhead is no larger than the specified o.

We can recast the bound in terms of data rates. Suppose an application has to support a minimum data rate of D packets per second. Under continuous transmission, this is equivalent to requiring that the maximum allowable overhead satisfies $o \geq 1 - D/W$ so that we may cast the expression for N_d in terms of the data rate D that the application has to support.

Observe that $N = N_d/(1-o)$ so that, in particular, the sender-side authentication latency τ to achieve confidence of at least $1-\delta$ of verifying received packets in a transmission group is approximately $N/W \approx N_d/W(1-o)$. Consider a low bandwidth, low computational capability example where the sender has access to a 1 Mbps channel, whence $W = 83$ packets per second assuming 1500 byte packets, and the receiver can only compute $K_s = 40$ public key checks per second. The hash/parity overhead is trivial as is easy to see: only a single hash packet is required. Consider a proportionate DoS signature attack with $R = 1$. Suppose the channel has an error rate of $p = 1/2$. With a choice of confidence parameter $\delta = 0.01$, and a maximum specified overhead of 20% the latency is approximately 2 seconds. If the maximum specified overhead is 5%, the latency is approximately 9 seconds. We provide some more examples focusing on higher bandwidths in the following sections. At 10-100Mbps we get latencies of 1-2 seconds and overheads of 1-3%.

Data flood attacks are easy to handle if the attack factor is moderate: we simply compute all hashes discarding packets that are spurious. If the attack factor exceeds the computational capability of the receiver to compute hashes, a data flood attack effectively increases the packet drop rate: as many arriving data packets are verified as there is computational capacity to handle the hashes. A variant of the bin approach may be used here as well to weed out large attacks on individual packets.

B.3 Acquisition Phase

We now derive a bound on the parameter g, the number of transmission groups, over which we buffer signature packets to protect against a replay attack. It is easy to see that the probability that we fail to acquire a valid current signature within g groups is γ^g where $\gamma = (1-(1-p)\pi)^{N_s}$. Thus choosing $g = \log\delta/\log\gamma$ suffices to ensure that we will obtain a valid current signature packet with confidence at least $1-\delta$.