Tradeoffs between Jamming Resilience and Communication Efficiency in Key Establishment*

Invited Paper

David Slater, Radha Poovendran, Patrick Tague
{*dmslater, rp3, tague*}*@u.washington.edu*
Network Security Lab (NSL), Dept. of Electrical Engineering
University of Washington, Seattle, WA, USA

Brian J. Matt
brian.matt@jhuapl.edu
Applied Physics Laboratory,
Johns Hopkins University, Laurel, MD, USA

We address the problem of allowing authorized users, who do not preshare a common key, to effectively exchange key establishment messages over an insecure channel in the presence of jamming and message insertion attacks. In this work, we jointly consider the security and efficiency of key exchange protocols, focusing on the interplay between message fragmentation, jamming resilience, and verification complexity for protocol optimization. Finally, we present three fragment verification schemes and demonstrate through analysis and simulation that in comparison with existing approaches, they can significantly decrease the amount of time required for key establishment without degrading the guaranteed level of security.

I. Introduction

In wireless networks, a principle necessity is the ability to securely and effectively communicate with intended receivers. Due to the broadcast nature of the wireless medium, however, a jamming adversary can send interfering signals to efficiently stop valid communication [18]. Extensive research has shown that mounting a jamming attack can be an effective denial-of-service (DoS) attack, with many proposed techniques available [20, 5, 8].

Numerous protocols have been presented for jamming mitigation, including direct-sequence spread spectrum (DSSS), frequency hopping spread spectrum (FHSS), and beamforming [13, 9]. These techniques rely on pre-shared secrets and specialized hardware, respectively, which are generally unavailable in a highly dynamic ad-hoc network. Uncoordinated frequency hoppping (UFH) was proposed for this scenario [16], where random frequency hopping techniques are used to avoid jamming activity. While UFH provides jamming resilience, it results in significantly lower throughput than key-based techniques, motivating its use for key exchange, after which keyed frequency hopping can be used.

In order to securely exchange keys in a manner that prevents adversarial spoofing attacks, the users must rely on a trusted-third party (TTP) for authorization. In an ad-hoc network, where the availability of the TTP is not guaranteed, certificates must given before deployment in the network. Thus, a key exchange between authorized users would require sending their certificates, public keys and private key contributions, resulting in a large key establishment message. Large messages have been shown to be critically vulnerable to reactive jamming attacks, where the adversary listens to the channel and starts jamming as soon as a signal is detected [19]. The message length is a critically important parameter in reactive jamming resilience, because a longer message gives the adversary more time to detect and effectively jam. Fragmenting the message allows greater resilience to reactive jamming, but at the cost of introducing a vulnerability to pollution attacks.

In a pollution attack [3, 7], an adversary inserts malicious fragments alongside the valid ones, resulting in decoding error. Prior to key exchange, this ne-

*A preliminary version of this work appears in the 2nd ACM Conference on Wireless Network Security (WiSec'09) [15].

This work is supported in part by the following grants: ONR YIP, N00014-04-1-0479; ARO PECASE, W911NF-05-1-0491; ARL CTA, DAAD19-01-2-001; and ARO MURI, W911NF-07-1-0287. This document was prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

cessitates a protocol to verify packet origins, without which an exponential search is required to find the valid subset of fragments. Protocols involving Merke-trees [17], hash trees used to determine invalid packets, and distillation codes [6], which allow a receiver to determine if a group of packets share the same origin, have been presented in the context of peer-to-peer (P2P) networks. In the seminal work [16] that showed the circular dependency between jamming mitigation techniques and key establishment, a fragment verification protocol was proposed which used a hash chain to verify packet origins, which we refer to as SPCC.

In this work, we analyze the tradeoffs between fragmentation overhead and reactive jamming vulnerability, and focus on optimizing the communication efficiency of the key exchange over the packet length. Furthermore, we seek to determine more efficient methods for fragment verification, in terms of communication overhead, while maintaining an equivalent level of security. We propose three candidate fragment verification protocols, and present an analytical basis for their communication efficiency, using SPCC as the baseline protocol.

The rest of the paper is organized as follows. In Section II, we present our communication and adversarial models and review the UFH protocol. We describe relevant background information in Section III. In Section IV, we present our three candidate protocols. In Section V, we analyze the verification protocols and fine-tune relevant parameters. Then in Section VI, we perform packet length optimization and compare the proposed protocols via a simulation study. We conclude in Section VII.

II. Communication & Adversary Models

In this section, we state our communication model, the necessary specifics of UFH, and our adversarial framework. Then we discuss the metrics used and define our security parameters. The notation used throughout this work is summarized in Table 1.

Table 1: A summary of notation used.

Symbol	Definition
c	number of orthogonal channels
d	time to switch between channels
m	message length (bits)
k	number of message fragments
l	packet length (bits)
r	packet frame information (bits)
p	probability of successful jamming
α	Rayleigh reactive jamming parameter
β	threshold of complete jamming resilience
γ	successful random jamming probability
F	successful reactive jamming probability
s	verification security level (bits)
T	average number of valid packets needed
t	average messsage completion time (bits)
Z	total number of packets received
$h(\cdot)$	hash function
H_k	k^{th} harmonic number, $H_k = \sum_{i=1}^{k} \frac{1}{i}$
(k, b)	erasure code mapping from k to b packets
q	probability of sending a header packet
w	witness (distillation codes)

II.A. Communication Model

We consider a sender and receiver in an ad-hoc wireless network attempting to securely and efficiently communicate, who have yet to obtain any shared information. They do, however, have certificates from an offline trusted third party, allowing them to authenticate key establishment messages from authorized users. They communicate over a set of c orthogonal channels, with each being able to simultaneously transmit on one channel and receive on another channel. We let d denote the number of bits that could be sent in the time taken to switch channels.

The sender's key establishment message of length m bits is broken into k fragments and each is inserted into a packet of length l bits. Each packet also includes all verification information and an additional r bits of frame information, including packet number. We assume that there is no general MAC protocol or feedback channel, due to the adversary's ability to monopolize those resources. Thus, the receiver cannot respond with acknowledgements (ACKs) or requests (REQs) for specific packets.

We use the UFH protocol for packet transmission, where the sender continuously cycles through all k packets, with each subsequent packet being sent across a random channel. Likewise, the receiver listens on random channels, but for sufficiently longer periods such that the time required to switch channels and the probability of truncating a packet by switch-

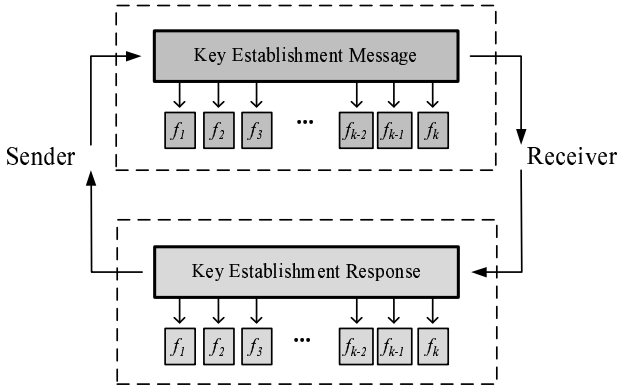

Figure 1: The logical exchange of key establishment messages between the sender and receiver requires only two message exchanges. Each message is disassembled into a collection of k fragments f_i.

ing can be neglected. This continues until a valid response has been received. Since ordinary ACKs are not available, a valid response is defined to be a full key establishment message response, which the receiver sends via UFH after decoding and authenticating the sender's message. Figure 1 illustrates a logical representation of the key exchange between the sender and receiver. One instance of a key establishment message is discussed in [16], where through an authenticated Diffie-Hellman protocol, the sender and the receiver includes its identity and public key, a signature of the public key, a timestamp, a contribution to the shared key, and a signature of the shared key contribution.

II.B. Adversarial Model

We consider an adversary capable of jamming packets both reactively and stochastically, and performing pollution attacks by broadcasting invalid packets. We assume that the energy required for the adversary to insert a packet is greater than that required to jam a packet, due to the observation that only a fraction of a packet needs to be jammed in order for the packet to be incorrectly decoded. Since we only consider protocols which have cryptographically secure fragment verification methods, where invalid packets are computationally efficient to identify, we assume that the adversary prefers solely to jam. Other adversarial models have considered the adversary as capable of randomly jamming packets with constant probability [16] or having a deterministic characterization of jamming where jamming always succeeds if sufficient power has been applied to it [8].

The adversary we consider is able to constantly jam a fraction $\gamma < 1$ of the available channels, since $\gamma = 1$ would never allow throughput, as in the case of a high-power wideband jammer. Since the sender and receiver are on random channels unknown to the adversary, the adversary's best response is to jam γc channels at random, which prevents the sender and receiver from having any recourse better than random. Thus, with a constant jamming adversary, the probability of the receiver correctly receiving a packet is $(1-\gamma)/c$, since the sender and receiver must randomly operate on the same channel. We assume that $c \gg 1$, ensuring that $(1-\gamma)/c$ is sufficiently small such that the probability of receiving multiple valid packets in the same cycle of k packets is negligible. The implication here is that each packet reception is independent of all others.

In addition to constant jamming, the adversary also listens for valid communication on the remaining channels, transmitting a high power pulse on a channel as soon as communication is detected on it. The effectiveness of this reactive jamming technique is primarily a function of the length of the packet sent, with a longer packet allowing more time for detection and a greater portion of the packet to be jammed. It is secondarily a function of the number of channels the adversary can eavesdrop on, the detection algorithm itself, the turnaround speed of the adversary's hardware, the geometric proximity of the parties, the coding rate, signal multipath, and others. In general, the curve mapping jamming power to probability of jamming success is sigmoidal [11]. Since there is linear correspondence between message length and reactive jamming duration, the resulting curve mapping message length to jamming success should also be sigmoidal. Thus, we represent the reactive jamming probability F as a shifted Rayleigh distribution,

$$F = u(l-\beta)\left(1 - e^{-\frac{(l-\beta)^2}{2\alpha^2}}\right), \qquad (1)$$

where $u(.)$ denotes the unit step function, l the length of the packet, α the Rayleigh parameter which measures sensitivity to jamming, and β the maximum packet length completely resilient to reactive jamming. This maximum packet length β is a function of the time taken by the adversary to detect and send out a jamming pulse, as well as the squared distance from the sender and receiver. Packets up to length β can only be randomly jammed, whereas when $l > \beta$, they are vulnerable to reactive jamming as well. Strictly speaking, $\alpha = 0$ does not result in a valid value for (1), so we take $\lim_{\alpha \to 0} F = 1 - u(\beta - l)$.

This causes a phase transition to a reactive jamming success probability of one for $l > \beta$. Likewise, for

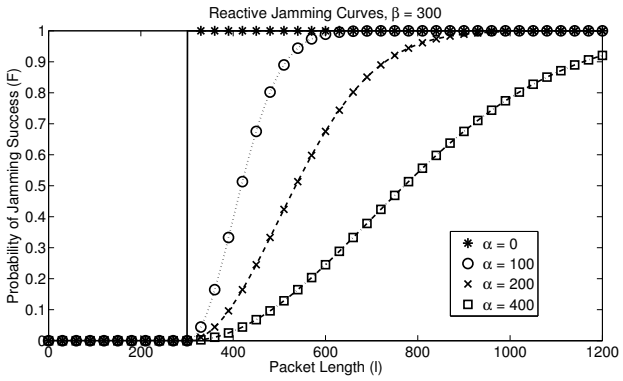

Figure 2: Reactive jamming distributions for various Rayleigh parameter choices, which maps the length of a packet to its probability of being reactively jammed, with $\beta = 300$ and $\alpha \in \{0, 100, 200, 400\}$.

$\alpha = \infty$ we take $\lim_{\alpha \to \infty} F = 0$, resulting in reactive jamming being impossible for any length. Figure 2 shows the reactive jamming distributions for various α values. Combining reactive and constant jamming leads to the jamming success rate

$$p = 1 - (1 - \gamma)\left(1 - u(l - \beta)\left(1 - e^{-\frac{(l-\beta)^2}{2\alpha^2}}\right)\right). \quad (2)$$

II.C. Security & Communication Efficiency Metrics

In comparing the advantages of our proposed protocols, we maintain a constant cryptographic strength, in terms of the computation required to directly perform cryptanalysis. We define the security level s, in bits, of a fragment verification protocol as the equivalent hardness of a pre-image attack on a length s cryptographic hash. We note that depending on the scheme used, the amount of cryptographic information may vary. For instance, while a hash function may only be length s, the DSA signature of equivalent security level has a length of $4s$ bits. We assume that the security level used is sufficiently large to prevent cryptanalysis-based pollution attacks. It is also assumed that Z key establishment authentications, where Z is the total number of received packets, is feasible for the receiver. This ensures that the receiver cannot be overwhelmed by complete key establishment messages, which would bypass fragment verification techniques.

We define the communication efficiency of a particular protocol as the expected amount of time until the receiver can verify a sufficient number of fragments to reconstruct the authentication message, divided by the bandwidth of the channel. Thus, the expected time t is measured in bits as

$$t = T(l + d)\frac{c}{1 - p}, \quad (3)$$

where T is the expected number of packets that need to be received in order to decode, $l + d$ is the amount of time spent sending a particular packet and changing channels, and $c/(1 - p)$ is the expected number of packets sent per received packet. This metric uses UFH as its communication model, and assumes packets are being sent continuously.

III. Background

We review preliminary background information that will form the technical groundwork for our proposed fragment verification techniques. We start by discussing Merkle trees and Distillation codes, which will be used for determining invalid fragments and determining packet originators. Then we will briefly discuss erasure coding techniques, which will be used to reduce redundancy in received packets.

III.A. Merkle Trees

A Merkle tree is a binary tree formed by hashes [10], which is used to quickly identify malicious fragments. For a group of k fragments, a Merkle tree of at most $2k - 1$ hashes can be formed for verification. Here, each fragment is hashed to form the leaf nodes of the tree. Then sibling nodes are paired, and their concatenation is hashed to form their parent. Nodes without a paired sibling are instead paired with an unpaired node at a higher level. This continues until the root of the tree, a single hash value, is formed.

When the k fragments are sent, the root of the verifying Merkle is sent also, with a signature identifying the sender of the root node. In order to verify the correctness of the entire message, the receiver duplicates the Merkle tree construction and compares the sent root with the constructed one. If these are unequal, then at least one of the k fragments must be invalid. To find the invalid packets, the receiver queries the source for intermediate nodes in the Merkle tree, which are then compared to the constructed nodes in order to verify respective subtrees. The sent intermediate nodes are shown to be part of the original Merkle tree by hashing them with their sibling to form their parent. Thus, invalid packets can be found by a binary search through the Merkle tree, though it requires a proper feedback channel.

III.B. Distillation Codes

A distillation code is a public-key based verification technique that allows the receiver to determine if a given pair of packets originated from the same source. This verification does not require knowledge of the sender's public key, but can rely instead on a globally known public key, allowing fragment verification to occur before message authentication.

Distillation codes are based on one-way accumulators [1, 2], which are two-to-one collision-resistant functions that for a sequence of elements returns the same result regardless of what order the elements were in the sequence. Accumulators can be used to generate a witness w for a fragment x that can verify that $x \in X$. We denote an accumulator as $h(u, x)$, mapping from $U \times X$ to U, with the following property, known as *quasi-commutativity*:

$$h(h(u, x_2), x_1) = h(h(u, x_1), x_2) \text{ for all } x_1, x_2 \in X \quad (4)$$

Thus, given an random nonce u and a set X, the returned value for all fragments $x \in X$ is identical, regardless of their order in X. We denote this accumulated value as $h(u, X)$. The reader can verify that the RSA based accumulator $h(u, x) = u^x \mod n$ in [1, 2] has the quasi-commutativity property.

The witness w for its associated fragment x can be used to verify what set of fragments it belongs to, with each set identifying a single sender. The necessary property for the witness of $x \in X$ is that $h(w, x) = h(u, X)$. To construct w_1 for $x_1 \in X$, the sender takes the set of fragments X, which must necessarily be chosen in advance, and chooses u arbitrarily. The sender then accumulates all fragments in X except x_1:

$$w_1 = h(u, X - x_1) = h(...h(h(u, x_2), x_3)..., x_k) \quad (5)$$

This is done similarly for all $x_i \in X$, so that:

$$h(w_i, x_i) = h(h(u, X - x_i), x_i) = h(u, X) \; \forall i \quad (6)$$

The receiver computes the value $h(w, x)$ for all packets received, and then partitions them into sets that share the same value. In order for the adversary to succeed in an insertion attack, it would be necessary to violate the collision-resistance property of the accumulator by finding u^* and x^* such that $h(u^*, x^*) = h(w, x)$. Thus, this is a valid fragment verification technique.

The bandwidth used by RSA based accumulators is at least the size of the modulus. Since reducing bandwidth is a key to achieving efficiency, we suggest Nguyen's accumulator scheme from bilinear pairings described in Section 4 of [12] for forming distillation codes. These accumulators add approximately $2s$ bits to the size of a fragment, which is significantly smaller than the RSA based alternatives.

III.C. Erasure Coding

Erasure coding [14] adds redundancy to messages in order to provide resilence to erroneous and missing fragments, requiring only a fraction of the resulting coded to be received correctly in order to successfully decode. A number of coding techniques have been presented, including both perfect codes such as Reed-Solomon [14], and near-optimal erasure codes such as Tornado coding [4]. A (k, b) erasure coding scheme takes k message fragments and adds $b - k$ redundant fragments, resulting in b coded fragments.

In perfect coding schemes, a total of k correctly received fragments are required for message reconstruction, at a computational complexity of $O(b^2)$. Near-optimal erasure codes have a decoding complexity of $O(b)$, but require $a + \epsilon$ correctly received fragments. Since a primary focus of this work is reducing the packet receptions necessary, we focus solely on perfect erasure coding, and ensure b is at a computationally feasible level. Erasure coding can only correct for known errors, so in order to utilize this technique, the receiver needs to be able to verify which received packets were valid.

IV. Verification Protocols

We present three approaches for efficient verification of message fragments. The first scheme focuses on minimizing communication time through reducing the packet space devoted to verification information. The purpose of the second scheme is to reduce communication time without impacting receiver or sender complexity. The final approach directly optimizes for communication time, keeping the complexity static even under adversarial attack.

IV.A. Hashcluster Scheme

We propose Hashcluster, which provides fragment verification by forming a hash chain from the message fragments. The message is split into clusters of n packets, with the n^{th} packet in each cluster containing the hash of the next n packets. The final cluster can have from 1 to n packets, with the final packet containing a hash of the entire message, as shown in Figure 3. In this scheme, each packet of length l includes r bits of frame information and a message fragment

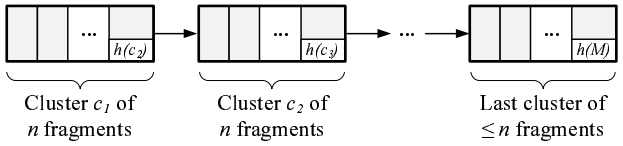

Figure 3: The Hashcluster scheme is illustrated for arbitrary sized clusters. The last packet in each cluster contains the hash of all packets in the next cluster, and the final cluster contains a hash of the entire message.

of up to $l - r$ bits, with every n^{th} packet containing a length s hash. The number of packets then, for a message of length m, is $k = \lceil m/(l - r - (s/n)) \rceil$. Since the message is fragmented among all of the packets in the chain, and a single missing packet will invalidate the verification method due to the hash structure, all packets are required for message reconstruction. The SPCC scheme, proposed in [16], corresponds to the case when $n = 1$.

In order for an adversary to modify the message fragment contained in a packet, they would be required either to perform a pre-image attack to match the modified message with the final hash, or to replace the final hash to match the modified message. Since pre-image attacks are cryptographically infeasible with the given security level, the former type of fragment insertion is impossible. However, since each cluster contains a hash of each subsequent cluster, it would be necessary to modify every cluster's hash prior to modifying the final packet. Thus, in a full received chain, either all packets originated from a single source, or all clusters were modified. In the worst case, the maximum number of key establishment authentications is when the adversary repeatedly modifies all clusters, which results in Zn/k authenications, which is considered feasible for $n \leq k$.

Since clusters are hashed instead of individual packets, it is necessary to attempt to verify every combination of packets in a cluster. To verify a hash cluster of length n packets, the receiver is forced to try a polynomial number of combinations of degree n, resulting in $(Z/n)^n$ hash operations in the worst case, where Z is the total number of packets received. In the absence of adversarial insertion, however, the number of hash operations reduces to $\lceil k/n \rceil$. Thus, the trade-off is a significant increase in receiver computational complexity during an insertion attack. For moderate values of n this becomes an unsurmountable challenge. Furthermore, as n grows, the fraction of space devoted to hash values decreases, thus diminishing the gain in further increasing n.

IV.B. Merkleleaf Scheme

We present Merkleleaf, a scheme where a partial Merkle tree is used to verify the data fragments. The aim of this approach is to increase communication efficiency without effecting sender or receiver complexity. Since the receiver is unable to query the sender for specific Merkle nodes and sending the entire tree would require significant overhead, our approach is to reduce the Merkle tree to the set of leaf nodes. The sender transmits the leaves of the Merkle tree in a header message and sends the key establishment information in a separate data message. Since both must be received to decode and respond, they are sent in conjunction with each other. The header message is broken into fragments and sent using the SPCC scheme, implying that all header fragments must be received in order to decode. With the header message received successfully, the hashes of all data packets are known and can therefore be verified in any order. Individual verification of data packets lends itself to erasure coding, which avoids the necessity of receiving all data fragments by only requiring a subset of coded fragment receptions. An overview of the scheme is shown in Figure 4.

The ordered triple (a, k, b) can be used to represent the Merkleleaf configuration, with a header packets sent by the SPCC scheme. The message data is then broken into $k = \lceil m/(l - r) \rceil$ fragments, and encoded using a (k, b) perfect erasure code. The size of b is bounded by $\lfloor (l - r - s)a/s \rfloor$, the size of the header message. Increasing the header message size allows for greater coding gain and therefore a reduction in redundancy, but also increases the number of needed packet receptions. The probability of sending a header packet is denoted by q, with a corresponding probability of $1 - q$ for a coded data packet.

IV.C. Witnesscode Scheme

We propose Witnesscode, an alternative approach based on distillation codes [6], which can be used to individually verify all packets without a header message. This allows erasure coding to be performed over all the packets and sets no limit on the number of coding fragments, which can therefore nearly eliminate the redundancy of received fragments. To frame this in the context of our communication model, the sender will generate a set of coded fragments to send, with associated witnesses computed for each of them. Each packet will include a single pair, allowing the receiver to independently verify each fragment received.

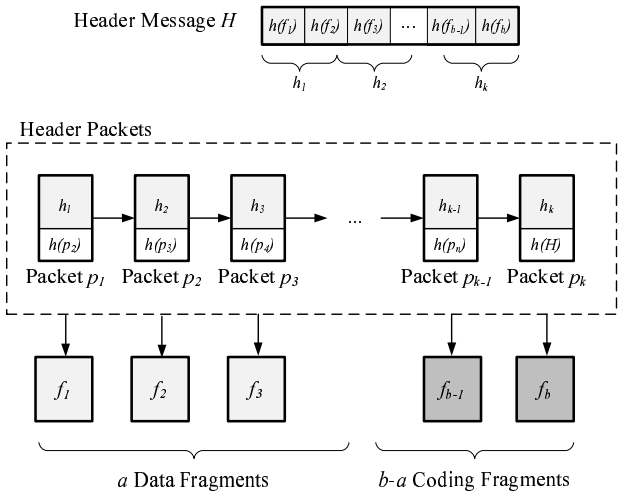

Figure 4: The Merkleleaf scheme is illustrated. The header packets are chained together using the SPCC scheme, forming a collection of hashes of the encoded data fragments. Upon collecting the header packets, the receiver can verify any encoded data fragment. The arrows between packets and fragments represents the relationships due to hash operations.

The sender generates a set of $k = \lceil m/(l-r-2s) \rceil$ data fragments, and encodes these with a (k, b) erasure code, resulting in b coded fragments. Then witnesses are formed for each at a cost of $b(b-1)$ one-way accumulator operations. Since this code is not constrained by a header message for coding length, the coding size is bounded only by the one-way accumulator computation at the sender and the Reed-Solomon decoding complexity of $O(b^2)$ at the receiver.

V. Analysis of Proposed Protocols

In order to analyze the parameters used for the proposed schemes prior to optimizing for packet length, we use the metric T, the expected number of received packets necessary for decoding. Throughout the analysis and simulations, we use a security level of $s = 112$ bits for packet validations, which results in 112 bit hashes and 224 bit witnesses. Modifying the security parameter would merely scale the relevant results, assuming symmetric hashes and bilinear pairing based accumulators, so it was kept constant throughout the analysis. We set the frame information to $r = 40$ and consider message sizes in the range $0 \leq m \leq 4000$ bits.

V.A. Hashcluster Analysis

We will first analyze the Hashcluster scheme, where we focus on reducing the storage space devoted to hashes, thus requiring fewer fragments and lowering received fragment redundancy. As shown by the authors of [16], requiring all packets to be received in conjunction with using UFH as a communication protocol introduces a significant amount of redundancy in received packets. While the receiver only needs k distinct valid fragments to decode the message, an expected number of kH_k fragments will be received, where

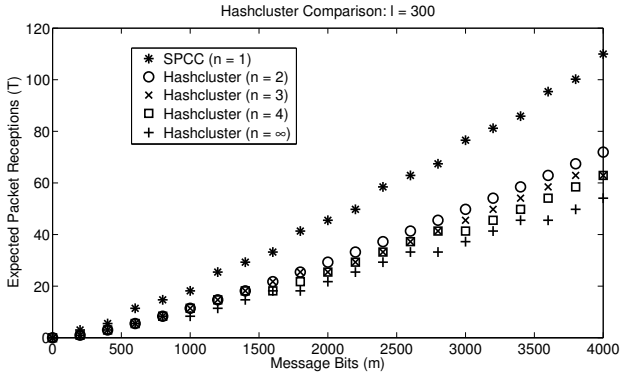

Figure 5: The average number of packet receptions for the Hashcluster scheme is evaluated as a function of the cluster size n, with the SPCC scheme representing the performance baseline of $n = 1$.

$$H_k = \sum_{i=1}^{k} \frac{1}{i}, \text{ the harmonic number of } k. \quad (7)$$

As $k \to \infty$ (with $\delta = 0.5772...$, the Euler-Mascheroni constant), this becomes

$$H_k \approx \ln(k) + \delta. \quad (8)$$

The gain in increasing n is in reducing the fraction of the packets devoted to hash information, thereby reducing the total number of fragments and the total transmit time. An auxiliary benefit of this is a reduction in the size of H_k and therefore the fraction of redundant packet receptions. Figure 5 shows $l = 300$ bits ($n = 1$ is the SPCC scheme, and $n = \infty$ corresponds to when the entire message is contained in a single cluster). Here, the worst case computation is $(Z/n)^n$ hash operations, where Z is the number of received packets. Since the most significant gain in efficiency is from $n = 1$ to $n = 2$, and raising n further gives little gain in efficiency but instead increases the complexity of the receiver by a polynomial degree at each step, we focus solely on the $n \in \{1, 2\}$ cases for the remainder of the simulation. We assume that $\frac{1}{4}Z^2$ operations is reasonable for the receiver.

V.B. Merkleleaf Analysis

For the Merkleleaf scheme, both the header message and the erasure coded data need to be received in order to decode. Since the header packets are combined using the SPCC scheme, the time to receive them is aH_a. In contrast, the coded data packets are received more efficiently. For a (k, b) code with k data packets and $(b - k)$ coding packets, only k packets need to be received in order to decode. Thus, the expected number of receptions is

$$\sum_{i=b-k+1}^{b} \frac{b}{i} = b(H_b - H_{b-k})), \tag{9}$$

which is significantly lower than kH_k. If we take $b \to \infty$, then by (8) and L'Hopital's rule, we get

$$\lim_{b\to\infty} b((\ln(b) + \delta) - (\ln(b - k) + \delta)) = k. \tag{10}$$

The expected number of packets T is then a combination of the expected header packet receptions and the expected data packet receptions. Specifically, the receiver must receive all of the k header packets and any a of the b erasure coded message packets. With probability q a header packet is sent, uniformly from the set of k header packets, and with probability $(1 - q)$, a data packet is sent, likewise uniformly from the set of b data packets. The joint expectation is derived in [15] as $T = g(0, 0)$, where $g(x)$ is given recursively by:

$$g(x) = \frac{1 + p(x, x_{i+})g(x_{i+}) + p(x, x_{j+})g(x_{j+})}{1 - p(x, x)} \tag{11}$$

$$g(i, j) = 0 \text{ when } i = k \text{ and } j \geq a \tag{12}$$

With the following definitions:

$$p(x, x_{i+}) = q\left(1 - \frac{i}{k}\right) \tag{13}$$

$$p(x, x_{j+}) = (1 - q)\left(1 - \frac{j}{b}\right) \tag{14}$$

$$p(x, x) = 1 - q\left(1 - \frac{i}{k}\right) - (1 - q)\left(1 - \frac{j}{b}\right) \tag{15}$$

The number of (a, k, b) triplet combinations is typically small, so the optimal allocation is found by searching all possibilities. The function T was experimentally determined to be convex in q. Therefore, for a given triplet, the optimal q was found using gradient descent methods.

For instance, consider $l = 376$ and $m = 2016$. This leaves space for two hash values in each header packet. Thus, if there are a header packets, the number of coded packets is $b = 2a$, with $k = 6$ data packets. The possible triplets (a, k, b) are $\{(3, 6, 6), (4, 6, 8), (5, 6, 10), (6, 6, 12), (7, 6, 14), (8, 6, 16)\}$, with (3, 6, 6) equivalent to the SPCC scheme due to a lack of coding. Values of $a \geq 9$ would be pointless, because the header message could contain the data message. Here, the optimal solution was (k, a, b) = (4, 6, 8) and $q = 0.461$.

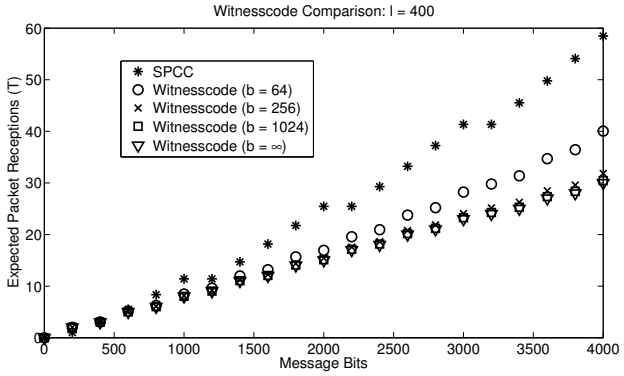

Figure 6: The average number of packet receptions for the Witnesscode scheme is evaluated for various coding values b, with the SPCC scheme included for reference. The case of $b = \infty$ represents the optimal coding case with infinite computational overhead.

V.C. Witnesscode Analysis

For the Witnesscode scheme, the coding parameter should be adjusted so that the computational complexities at the sender and receiver are still feasible. As shown in Section IV.C, the sender and receiver have a computational complexity of $O(b^2)$, where b is the total number of coded packets. To determine an adequate tradeoff between computation and communication, we show the results of several values of b in Figure 6, with $l = 400$. The SPCC scheme is given alongside for reference.

As can be seen in Figure 6, the Witnesscode points quickly converge to the optimal for modest values of b. For the remainder of this work, when we discuss and simulate this scheme, it is in reference to the $b = 256$ case, because it is extremely close to the infinite case in terms of communication efficiency, and has a reasonable computational complexity.

VI. Performance Evaluation

We present our packet length optimization techniques and provide a simulation study to evalutate the proposed schemes. For the following simulations, we let $s = 112$, $r = 40$, and $d = 64$.

VI.A. Packet Length Optimization

In order to provide thorough evaluation of the candidate schemes and comparison with respect to the SPCC scheme, we use the metric based on the expected number of bits sent, defined in (3). Finding the optimal values of l through taking the derivative of t, we note that t is proportional to $(1-\gamma)/c$, implying that γ and c have no effect on this optimization. Thus, for simplicity we scale t by $c/(1-\gamma)$.

$$\begin{aligned} &\min_l T(l+d)\frac{c}{1-p} \\ &t = T(l+d)/\left(1-u(l-\beta)\left(1-e^{-\frac{(l-\beta)^2}{2\alpha^2}}\right)\right) \end{aligned} \tag{16}$$

By evaluating the two cases of the unit step function this becomes

$$t = \begin{cases} T(l+d)e^{\frac{(l-\beta)^2}{2\alpha^2}} & \text{if } l > \beta \\ T(l+d) & \text{if } l \leq \beta \end{cases} \tag{17}$$

Depending on the jamming characteristics, the communication efficiency can be dramatically improved by using longer packet lengths. While larger packets result in a higher sensitivity to reactive jamming techniques, doing so reduces the number of message fragments. This in turn lowers the number of expected packet receptions T, and in addition reduces the impact of frame information and time spent switching channels. On the other hand, drastically increasing packet length will result in exponentially diminishing throughput. The gain in optimizing for packet length becomes more pronounced as α increases, relative to β. When $\beta \gg \alpha$ the gain from increasing l beyond that of β is essentially zero. This is especially seen in the case when $\alpha = 0$, where any packet longer than β is jammed with probability one.

In general, a closed-form solution for the optimal value of l for a particular scheme is not expressible. However, for certain conditions on the parameters, these equations can be simplified.

For the Witnesscode scheme, if we take b and m as large, then by (10) the expected number of packets can be approximated as $T = k = \frac{m}{l-r-2s}$. In order to find the optimal value of l in t, insert this value of T into (17) and solve for l.

$$\frac{d}{dl}\frac{l+d}{(l-r-2s)(1-F)} = 0 \tag{18}$$

If $l < \beta$, then by (17):

$$\frac{d}{dl}\frac{l+d}{l-r-2s} = \frac{-r-2s-d}{(l-r-2s)^2} = 0 \tag{19}$$

This implies that the optimal value of l is not found in this region, with the possible exception of the endpoint $l = \beta$. For the second case, $l \geq \beta$, we have:

$$\frac{d}{dl}\frac{l+d}{l-r-2s}e^{(l-\beta)^2/2\alpha^2} = 0 \tag{20}$$

Using the product rule for the exponential term and simplifying, we get:

$$\begin{aligned} -(d-r-\beta-2s)l^2 + (-\beta d + (\beta-d)(r+2s))l \\ + l^3 - (\beta d(r+2s) - (r+2s+d)\alpha^2) = 0 \end{aligned} \tag{21}$$

This can then be solved using the cubic equation. Then, given (19) and (21), when a real root exists to the cubic equation and is $> \beta$ it is the asymptotic limit. Otherwise the the limit is the endpoint, $l = \beta$.

In the case of the Hashcluster and SPCC schemes, on the other hand, when m is large, from (8), we get:

$$T = \left(\frac{m}{l-r-s/n}\right)\left(\ln\left(\frac{m}{l-r-s/n}\right)+\delta\right) \tag{22}$$

This contains a logarithmic term that cannot be reduced further or canceled through differentiation, leaving an equation involving polynomial and logarithmic terms in l. Therefore, we must rely on numeric methods for computation. Figure 7 shows the general shape of packet length curves for the SPCC scheme with $m = 2000$ and $\beta = 300$. Since the curves are nearly convex, with several piecewise convex sections, it is possible to efficiently optimize over l using numeric methods.

We now look at the results from optimizing over the packet length for the various protocols. We vary the parameter α, with $\alpha = 0$ as a baseline, representing the case where packet sizes are at or below the jamming threshold β, and thus are sufficiently small to provide immunity to reactive jamming. First, we show the effect that increasing α has on the t versus l curves for the SPCC case. Figure 7 shows the time taken by various packet lengths, for different jamming parameters. As can be seen, the optimal l value for each of these curves (resulting in the min value for the corresponding t), increases with α.

Finally, we look at the impact that packet length optimization has on the communication efficiency, for different jamming characteristics, with $\alpha \in \{0, 100, 200, 400\}$ and $\beta = 300$. Figure 8 shows the effect of this optimization. As can be seen, this can significantly reduce the time spent for communication. For the SPCC scheme, the $\alpha = 400$ takes nearly

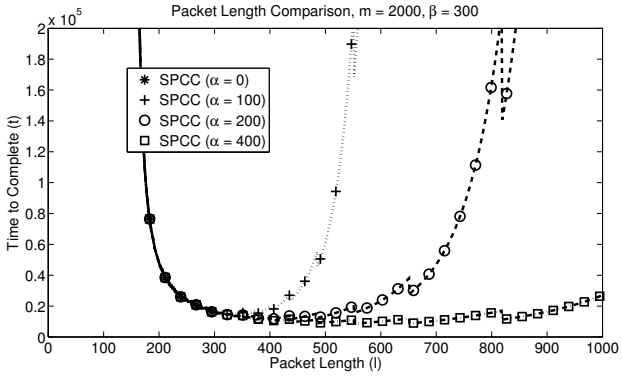

Figure 7: Tradeoff between fragment verification and reactive jamming resilience, mapping packet length l to message exchange time t, with $m = 2000$, $\beta = 300$, and $\alpha \in \{0, 100, 200, 400\}$.

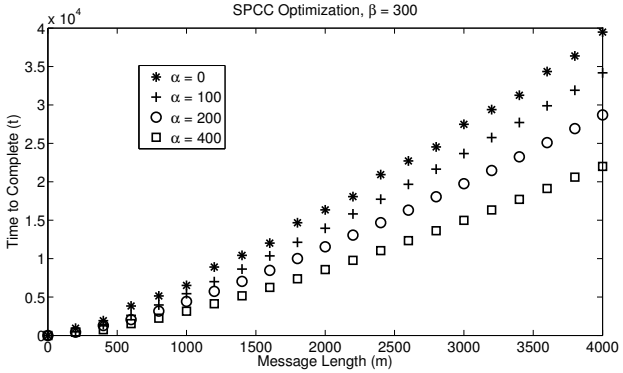

Figure 8: Time t to complete the message exchange after performing packet length optimization on the SPCC scheme. α is varied among 0, 100, 200, 400 and $\beta = 300$. The $\alpha = 0$ case represents the performance baseline of the optimization.

half the time as $\alpha = 0$, indicating that under variable reactive jamming conditions this optimization can result in a significant gain in communication efficiency.

VI.B. Protocol Comparison

Next we compare the SPCC, Hashcluster ($n = 2$), Merkleleaf, and Witnesscode ($b = 256$) schemes, with their parameters as derived in Section V. We show $(\alpha, \beta) \in \{(0, 300), (100, 300), (400, 400)\}$ in Figures 9, 10, and 11, respectively.

To describe the benefits to each scheme, we begin with SPCC. First, as is clear from the preceding figures, when the message is small (less than 1000 bits), the SPCC scheme has an efficiency roughly equivalent to the other three schemes. Since it is the simplest to implement and has the least computational requirements for the sender and receiver, it would be preferable in this case. The reason for the SPCC scheme performing well for small messages is that the number

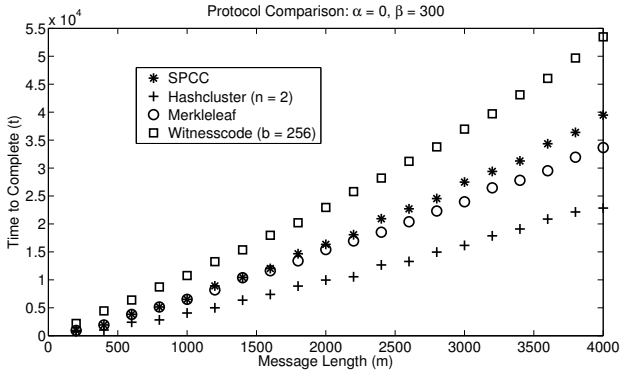

Figure 9: The expected time t to complete a message exchange for all four protocols, with $\alpha = 0$ and $\beta = 300$. The Witnesscode scheme does significantly worse than the other schemes and Hashcluster does significantly better, due to their respective fraction of data spent on verification.

of packets is likewise small, thus reducing the probability of receiving additional redundant packets.

The Hashcluster scheme is fairly computationally challenging, though this computation is only done in the presence of an adversary inserting erroneous packets. During normal operation, or in the presence of a jamming adversary, only the non-redundant packets received need to be verified. Thus, as long as the verification is computationally feasible during insertion attacks, the adversary will choose to spend its energy jamming instead, and the computation will never need to be done. The Hashcluster scheme performs especially well for small values of α and β. In this constrained environment, the small fraction of payload data devoted to verification frees up a significant portion of the packets for message data.

The Merkleleaf scheme out-performs the SPCC scheme for larger values of m, though it has a similar verification complexity. The only additional computation of note is the Reed-Solomon coding on the data packets. However, since the code length b takes small values for all m considered, the $O(b^2)$ decoding operation is insignificant. Thus, when it is computationally infeasible to use the Hashcluster or Witnesscode schemes, this becomes the optimal approach. For small values of m (less than about 1500), the Merkleleaf scheme simplifies to the SPCC scheme, where only the header message is sent. For large values of m, α, and β, it can perform equivalently to the Hashcluster scheme.

Finally, the Witnesscode scheme has its greatest benefit for large m, α, and β. Due to the witness being twice the size of a typical hash, small values of α and β constrain the fragment size, greatly limiting the

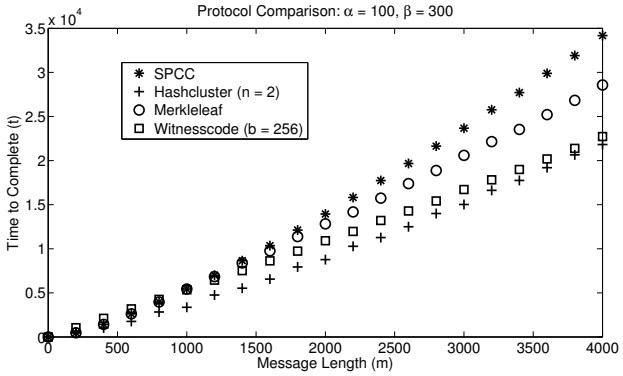

Figure 10: The expected time t to complete a message exchange for all four protocols, with $\alpha = 100$ and $\beta = 300$. The SPCC and Merkleleaf schemes are behave similarly for $m < 1500$, after which they diverge due the coding gain in the Merkleleaf scheme.

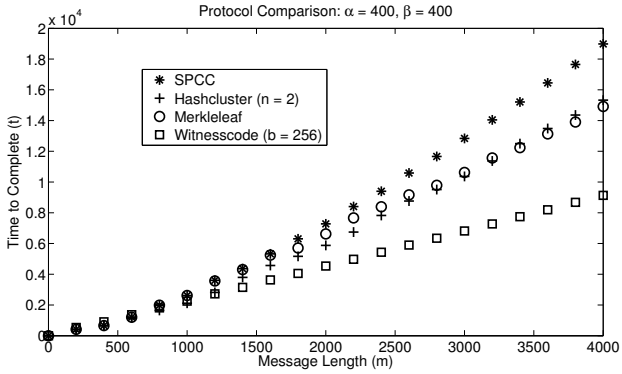

Figure 11: The expected time t to complete a message exchange for all four protocols, with $\alpha = 400$ and $\beta = 400$. The Witnesscode greatly outperforms the other protocols for $m > 1200$, due to a large coding gain, and the Merkleaf and Hashcluster schemes perform similarly for all m.

fraction of the packet payload devoted to the message. On the other hand, a large m implies a large number of packets, which was where the inefficiency in the first three schemes was located. In this case, however, the gain from coding is significant, because the probability of getting a redundant packet is reduced to near zero. On the other hand, the computational complexity is non-trivial and must be performed regardless of whether the adversary is or is not inserting packets.

VII. Conclusion

In this work, we addressed the problem of efficient key exchange between authorized users in an insecure wireless network. We considered an adversary capable of performing both reactive jamming and message insertion attacks. Our approach was to optimize for the communication efficiency of the key exchange while maintaining the desired level of security, focusing on the tradeoff between resilience to reactive jamming and fragment verification complexity. We then proposed three verification schemes to further improve communication efficiency which can be tailored to specific network scenarios: Hashcluster, Merkleleaf, and Witnesscode. Future work will address the problem of outperforming a random communication model in jamming avoidance prior to key exchange.

References

[1] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. *Advances in Cryptology – EUROCRYPT '97*, pages 480–494, 1997.

[2] J. Benaloh and M. de Mare. One-way accumulators: a decentralized alternative to digital signatures. *Advances in Cryptology – EUROCRYPT '93, Proc. of the Workshop on the Theory and Applications of Cryptographic Techniques*, pages 274–285, 1994.

[3] L. Buttyán, L. Czap, and I. Vajda. Securing coding based distributed storage in wireless sensor networks. In *IEEE Workshop on Wireless and Sensor Network Security (WSNS)*, Atlanta, GA, USA, Sept. 2008.

[4] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, 28(4):56–67, 1998.

[5] V. Gupta, S. Krishnamurthy, and M. Faloutsos. Denial of service attacks at the MAC layer in wireless ad hoc networks. *Military Communications Conference (MILCOM 2002)*, 2:1118–1123, 2002.

[6] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. D. Tygar. Distillation codes and applications to DoS resistant multicast authenication. In *The 11th Annual Network and Distributed System Security Symposium (NDSS 2004)*, San Diego, CA, USA, Feb. 2004.

[7] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in P2P file sharing systems. *Proc. IEEE 24th Annual Joint Conference of the IEEE*

Computer and Communications Societies (INFOCOM 2005), 2, 2005.

[8] G. Lin and G. Noubir. On link layer denial of service in data wireless lans. *Wireless Communications and Mobile Computing*, 5(3):273–284, May 2005.

[9] W.-T. Lin and K.-B. Yu. Adaptive beamforming for wideband jamming cancellation. *IEEE National Radar Conference*, pages 82–87, 1997.

[10] R. Merkle. Protocols for public key cryptosystems. In *Proc. 1980 IEEE Symposium on Security and Privacy (S&P '80)*, pages 150–159, Apr. 1980.

[11] F. Meshkati, H. Poor, S. Schwartz, and N. Mandayam. An energy-efficient approach to power control and receiver design in wireless data networks. pages 1885–1894, 2005.

[12] L. Nguyen. Accumulators from bilinear pairings and applications. *Topics in Cryptography - CT-RSA 2005*, pages 275–292, 2005.

[13] R. A. Poisel. *Modern Communication Jamming Principles and Techniques*. Artech House, 2004.

[14] R. M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[15] D. Slater, R. Poovendran, P. Tague, and B. J. Matt. A coding-theoretic approach for efficient message verification over insecure channels. In *Second ACM Conference on Wireless Network Security (WiSec '09)*, Zurich, Switzerland, Mar. 2009.

[16] M. Strasser, C. Pöpper, S. Čapkun, and M. Čagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proc. 2008 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2008.

[17] C. Wong and S. Lam. Digital signatures for flows and multicasts. In *Proc. on the 6th International Conference on Network Protocols (ICNP '98)*, pages 198–209, Oct. 1998.

[18] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, Oct. 2002.

[19] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: Attack and defense strategies. *IEEE Network*, 20(3):41–47, May/June 2006.

[20] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proc. of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 46–57, 2005.