# A Coding-Theoretic Approach for Efficient Message Verification Over Insecure Channels*

David Slater*, Patrick Tague*, Radha Poovendran*, Brian J. Matt†
*Network Security Lab (NSL), Dept. of Electrical Engineering
University of Washington, Seattle, WA, USA
†Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, USA
{dmslater, tague, rp3}@u.washington.edu, brian.matt@jhuapl.edu

## ABSTRACT

We address the problem of allowing authorized users, who have yet to establish a secret key, to *securely* and *efficiently* exchange key establishment messages over an insecure channel in the presence of jamming and message insertion attacks. This problem was first introduced by Strasser, Pöpper, Čapkun, and Čagalj in their recent work, leaving joint consideration of security and efficiency as an open problem. In this paper, we present three approaches based on coding theory which reduce the overall time required to verify the packets and reconstruct the original message in the presence of jamming and malicious insertion. We first present the Hashcluster scheme which reduces the total overhead included in the short packets. We next present the Merkle-leaf scheme which uses erasure coding to reduce the average number of packet receptions required to reconstruct the message. We then present the Witnesscode scheme which uses one-way accumulators to individually verify packets and reduce redundancy. We demonstrate through analysis and simulation that our candidate protocols can significantly decrease the amount of time required for key establishment in comparison to existing approaches without degrading the guaranteed level of security.

*This work is supported in part by the following grants: ARO MURI, W911NF-07-1-0287; ARO PECASE, W911NF-05-1-0491; and ARL CTA, DAAD19-01-2-001. 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.


*WiSec'09,* March 16–18, 2009, Zurich, Switzerland.


## Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*security*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols

## General Terms

Algorithms, Performance, Reliability, Security

## Keywords

Insecure channels, message verification, jamming, insertion attacks

## 1. INTRODUCTION

Wireless network communications are critically vulnerable to denial-of-service (DoS) attacks [1]. Due to the broadcast nature of the wireless channel, an adversary can transmit signals to interfere with those of authorized users, thus jamming valid communication. With a variety of methods to choose from (constant, reactive, pulse, etc.) [27], extensive research has shown that mounting jamming attacks is straightforward and resource efficient [1, 25]. Additionally, an adversary can utilize cross-layer information to perform more devastating jamming attacks by targeting specific layers (MAC, link, network) [8, 12]. Alternatively, an adversary can maliciously insert packets into the wireless channel, often referred to as a pollution attack [4, 11], causing erroneous message reception and reducing throughput.

In response to jamming, numerous mitigating protocols have been proposed, such as spread spectrum and beamforming. The latter method uses specialized hardware (directed antennas) and the geometric distinction of senders to distinguish between jamming and valid signals [13]. Direct-sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS) [18, 20] take advantage of prior shared secrets between the sender and receiver to minimize adversarial interference, forcing the adversary to send interfering signals randomly. The reliance on shared secret information for jamming mitigation requires security associations to be formed between the communicating parties before the adversary's arrival or through a secure channel. In ad-hoc networks, where adversaries may be present prior to deployment, topology and node associations are highly dynamic, and specialized hardware may not be available, these mitigation techniques are not suitable.

In order to mitigate the effects of pollution attacks, the receiver must be able to verify the source of individual messages. However, the exchange of lengthy messages, such as those including public-key signatures, is critically vulnerable to reactive jamming attacks in which the adversary starts jamming once a packet transmission is overheard [26]. Hence, there is a direct conflict between mitigation of pollution attacks and resilience to reactive jamming. Without cryptographic methods to verify the packets' origins, such as Merkle trees [24] and distillation codes [10], the receiver cannot distinguish between valid and maliciously inserted packets. This results in an exponential number of computations to find the subset of valid fragments. Furthermore, the inability to verify packet origins invalidates receiver feedback channels. With these added difficulties, it becomes impossible to perform a typical key exchange.

In the seminal work of Strasser, Pöpper, Čapkun, and Čagalj [23], the circular dependence between secure key exchange and DoS mitigation was first noted. They proposed a communication protocol called Uncoordinated Frequency Hopping (UFH), where the communicating parties listen and send on randomly selected orthogonal channels, causing random jamming to be the optimal adversarial response. Here, a message is broken into fragments, which are repeatedly sent across random channels. They demonstrate that even if the adversary can jam a large fraction of channels, the communicating parties are still able to achieve a bandwidth inversely proportional to the probability of successful jamming. In addition to the UFH communication model, the authors proposed a secure fragment verification scheme, henceforth referred to as SPCC. In this scheme, packets are logically connected forming a hash chain, where each packet includes the hash of the next packet, and the final packet includes a hash of the first message fragment. This allows verification of individual packets, lowering the computational complexity from exponential to linear, and ensuring that a valid chain must originate from a single source.

We address the same problem of key establishment from a performance standpoint, with the intent of improving the communication efficiency of fragment verification while maintaining the security of the scheme. We employ the same framework, using the UFH protocol for our communication model and the SPCC scheme as a performance benchmark. We propose three candidate schemes for improved efficiency, with tradeoffs based on various metrics. Specifically, we are interested in the communication time, sender complexity, and receiver complexity both in the presence and absence of an adversary. We then present an analytical basis for our results, and illustrate their tradeoffs through a simulation study.

The remainder of the paper is organized as follows. In Section 2, we revisit the UFH protocol and SPCC scheme, and present our communication and adversarial models. We review prerequisite material in Section 3. In Section 4, we present candidate protocols. Then in Section 5, we analyze the efficiency of proposed protocols and compare to previous work. We conclude in Section 6.

# 2. COMMUNICATION & ADVERSARY MODELS

As we are utilizing the same communication framework as given in the seminal work [23], we will review the particulars of the UFH protocol, revisit the SPCC scheme, and state our additional assumptions. The notation used throughout this work is summarized in Table 1.

**Table 1: A summary of notation used.**

| Symbol | Definition |
|---|---|
| $c$ | number of orthogonal communication channels |
| $p$ | probability of successful jamming |
| $m$ | message length (bits) |
| $l$ | packet payload (bits) |
| $s$ | fragment verification security strength (bits) |
| $k$ | number of message fragments |
| $V$ | number of sender's (valid) packet receptions |
| $Z$ | number of total packet receptions |
| $t$ | normalized time, average packet reception time |
| $h(\cdot)$ | hash function |
| $H_k$ | $k^{th}$ harmonic number, $H_k = \sum_{i=1}^{k} \frac{1}{i}$ |
| $(a, b)$ | erasure code mapping $a$ data to $b$ coded packets |
| $q$ | probability of sending a header packet (Merkle) |
| $w$ | witness (distillation codes) |

## 2.1 Communication Model: UFH

We make the following assumptions, as in [23]. We consider a sender and a receiver attempting to communicate with each other using a set of $c$ orthogonal channels. We assume that each user has the ability to transmit on one set of channels and listen on a different set of channels, assuming the time required to switch between channels is negligible. For this work we consider the case where the size of each set is one. The communicating parties have no shared secrets before key establishment, nor do they have knowledge of each other's public keys, but they do have certificates from a trusted authority, allowing them to authenticate key establishment messages from valid parties.

Messages of $m$ bits originating at the sender are broken into $k$ fragments and placed in $k$ packets of size $l$ bits, each labeled with a fragment number to reduce complexity of message reconstruction. We assume that $l$ is sufficiently small to provide immunity to reactive jamming and that the adversary cannot intelligently modify packet contents by jamming precise bits. In UFH, the sender continually cycles through the $k$ packets, sending them on randomly selected channels. Meanwhile, the receiver listens on random channels for long periods of time, where we assume that the probability of switching channels during a packet reception is zero. To compensate for lost packets, the sender continues re-sending all fragments until the entire response message can be decoded. Instead of sending acknowledgments (ACKs) or requests (REQs) for individual fragments, as they would be indistinguishable from those sent by the adversary over the un-authenticated channel, the receiver sends an authenticated ACK implicitly through the key establishment response. This is done using the same UFH technique once the initial message has been received, reconstructed, and authenticated. The logical connection between the sender and receiver in this exchange of key establishment messages is illustrated in Figure 1. An example of the key establish-

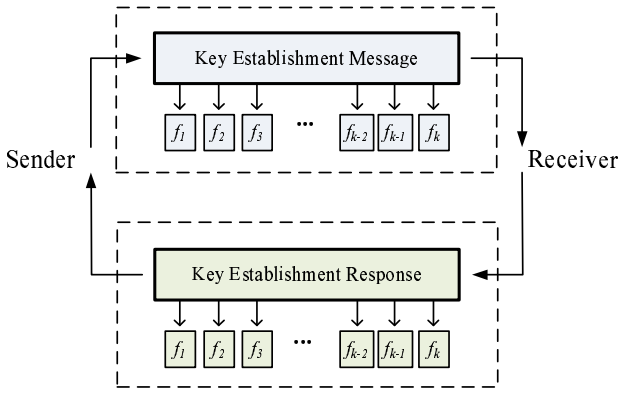


**Figure 1: The logical exchange of key establishment messages between the sender and receiver requires only two message exchanges. However, each message is disassembled into a collection of $k$ fragments $f_i$ of small enough size to provide resilience to reactive jamming attacks.**

ment message is the authenticated Diffie-Hellman protocol discussed in [23] in which each of 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.

The adversary has the ability to constantly jam up to a fraction $p$ of the $c$ channels, which if randomized effectively jams each packet with probability $p$. We assume $p < 1$, as communication is impossible if $p = 1$, as in the case of constant, wideband jamming. Alternatively, the adversary can insert a large number of malicious packets into communication channels. We assume that the cost of adversarial insertion of a packet is greater than the cost of jamming a packet. As a consequence, we assume that the adversary prefers to solely jam whenever the receiver is computationally capable of detecting and eliminating packet insertion.

From the adversary's perspective, communication via UFH is identical to key-based frequency hopping. This reduces the adversary's best response to random jamming, which for $p < 1$ always allows a positive throughput. Specifically, the probability of correctly receiving a message is then $(1-p)/c$. We assume that the number of channels $c$ is sufficiently large such that the probability of collisions between the sender's transmissions and receiver's replies is negligible. Finally, we assume that $(1 - p)/c$ is sufficiently small, such that the probability of receiving multiple packets on the same cycle of $k$ packets is negligible, and thus each packet reception is independent of the others.

## 2.2 SPCC Scheme, Revisited

The SPCC scheme is presented in [23], where a hash chain is used to verify packets originated from the same source. By including the (length $s$) hash of each packet in the preceding packet, the adversary is prevented from inserting malicious packets in the chain after valid packets, greatly simplifying the verification complexity. Additionally, the final packet contains a hash of the first message, disabling the adversary's ability to insert malicious header packets. Thus, through this circular method, the entire chain can be verified as originating from the same source.

In this scheme, the message of length $m$ is broken into $k$ fragments based on the packet size $l$, given by $k = \lceil m/(l - s)\rceil$ (assuming $l > s$). Thus, packets consist of a fragment of length $(l - s)$ and a hash of length $s$. Since the message is fragmented among all packets in the chain, the reception of all fragments is necessary for verification.

The maximum verification computation for this scheme occurs at the receiver, and is equal to $Z$ hash operations in the worst-case, where $Z$ is the total number of packet receptions. In parallel to this, the maximum number of message authentications at the receiver is $\lfloor Z/k \rfloor$, corresponding to the case when all packets are distinct and form complete hash chains.

## 2.3 Communication Efficiency

In order to prevent pollution attacks, fragments are verified with cryptographic functions of security level $s$ bits. We assume that $s$ is sufficiently large to prevent cryptanalysis. We note that the depending on the scheme used, the actual length of the cryptographic block may vary. For instance, while a hash of length $s$ bits has security level $s$ against pre-image attacks, the RSA signature of equivalent security level has a length of $4s$ bits.

Using UFH as our communication model, we measure the communication efficiency of a given scheme according to the time it takes for the receiver to receive a sufficient number of fragments and decode the key establishment message. We normalize this by the average time between packet receptions, so that the time $t$ to complete a message transmission equals the expected number of valid packet receptions $E[V]$ needed to correctly decode.

# 3. BACKGROUND

Here, we present relevant background information that will form the basis of the protocols we present. Applying techniques from pollution attack mitigation in peer-to-peer (P2P) networks and recasting them in the current context, we extract the following tools: erasure coding to correct for missing fragments, Merkle trees to identify incorrect fragments, and distillation codes to verify the origin of each fragment.

## 3.1 Erasure Coding

Erasure coding [21] adds redundancy to a message in order to provide resilience to erroneous and missing fragments, thus requiring only a fraction of the coded message needing to be received correctly. An $(a, b)$ (perfect) erasure code takes $a$ data symbols and adds $(b - a)$ coding symbols, for a total of $b$ symbols. In order for original message to be decoded, a total of $a$ symbols need to be received (the same as in the uncoded case). Since erasure coding can only correct for missing symbols in known locations, maliciously inserted or out-of-order fragments will result in decoding errors. Hence, the order of packets must be verified before decoding.

Perfect erasure coding, such as Reed-Solomon coding [21], is fairly expensive, though still manageable for moderate values of $b$, at $O(b^2)$ operations to decode. Near-optimal erasure coding, such as Tornado coding [5], Raptor coding [22], or LT coding [14], is a linear-time operation ($O(b)$), but requires $a+\epsilon$ symbol receptions. Since the focus of this work is communication efficiency, we use perfect erasure coding for our protocols.

## 3.2 Merkle Trees

A Merkle tree [16] is a binary tree of hash values, containing no more than $2k$ hashes for a collection of $k$ fragments. The purpose of the tree is to efficiently verify the message fragments. The leaf nodes of the tree are given by the hash values of their respective fragments. Then neighboring pairs of nodes are hashed together to form their parent nodes, which is repeated until a single root node is left, which is signed by the originator. In order to determine the validity of the message, it is only necessary to verify that the root node received matches the corresponding hash of the message. However, if these do not match, the receiver can determine erroneous fragments through a binary search of the Merkle tree.

## 3.3 Distillation Codes and One-Way Accumulators

A distillation code [10] can be used to verify if a group of packets share the same origin. This verification step does not require knowledge of the sender's public key, allowing this verification to take place prior to message authentication. The cryptographic overhead used by a distillation code is equivalent to twice the size of the private key used, as opposed to four times the number of security bits, in the case of RSA and ECDSA signatures. In this work we consider ECDSA, which has a significantly smaller public key than RSA [17]. Distillation codes are based on one-way accumulators, of which we will review the necessary properties.

An accumulator [2, 3] is a two-to-one collision-resistant hash function $h(u, x)$, mapping from $U \times X$ to $U$, with the property

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

known as *quasi-commutativity*. The implication of quasi-commutativity is that given an arbitrary starting value $u$ and a set $X$ of fragments, the result of sequentially hashing all fragments $x \in X$ is the same regardless of the order they were hashed in. In other words, it is the accumulated hash of the set $X$. An example hash function which demonstrates quasi-commutativity is the RSA accumulator $h(u, x) = u^x \mod n$.

Accumulators can be used to generate a witness $w$ for a fragment $x$ that can verify $x \in X$, with $X$ being the set of fragments originated from a particular source. To construct the witnesses $w_1$ for a fragment $x_1 \in X$, the sender picks a random $u$ and computes

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

accumulating all messages in $X$ except for $x_1$. This is repeated for all of the remaining messages in $X$, resulting in $k(k-1)$ hash operations. In verifying the witness,

$$h(w_i, x_i) = h(h(u, X - x_i), x_i) = h(u, X)$$

for all $i$. In order to determine two fragments $x_i$ and $x_j$ belong to the same set $X$, the receiver verifies $h(w_i, x_i) = h(w_j, x_j)$. In order for an adversary to insert a malicious fragment into a set, it is necessary to find a collision $h(u^*, x^*) = h(u, X)$, which would violate the collision-resistance property.

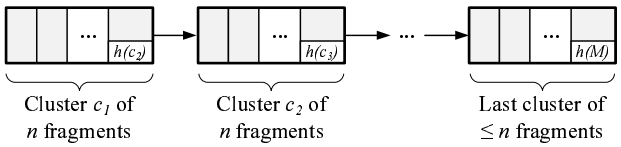


**Figure 2: The Hashcluster scheme is illustrated for arbitrary sized clusters. The last packet in each cluster contains the hash of all packets in the next cluster, and the final cluster contains a hash of the entire message.**

# 4. VERIFICATION PROTOCOLS

Since the SPCC scheme requires all fragments in order to decode and fragments are received randomly, there is a high degree of redundancy in received fragments, as the authors pointed out in Section 3.4 of [23]. While the receiver only needs $k$ fragments to authenticate the message, the receiver will on average receive $kH_k$ fragments, where $H_k$ is the harmonic number of $k$, that scales as $k \log_2(k)$. Thus, for even a moderate value like $k = 11$, the average number of received fragments $t$ is greater than $3k$. Our aim is to optimize for communication efficiency, while maintaining the level of security guaranteed by the SPCC scheme.

Here, we present three approaches for efficient verification of message fragments. The first scheme focuses on minimizing communication time through reducing the necessary number of fragments, without impacting sender complexity. 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.

## 4.1 Hashcluster Scheme

We propose Hashcluster, a generalization of the SPCC scheme, where blocks of packets are hashed instead of individual packets, as shown in Figure 2. Thus, the $n^{th}$ packet in each cluster will contain the hash of the next $n$ packets, which will likewise contain the hash of the following $n$ packets. The final packet contains the hash of the entire message. The number of packets is then given by $k = \lceil m/(l-(s/n)) \rceil$. The gain 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. Like the SPCC scheme, all packets need to be received for decoding.

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. 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 differential gain in increasing $n$. As in the SPCC scheme, the maximum number of authentications is $\lfloor Z/k \rfloor$.

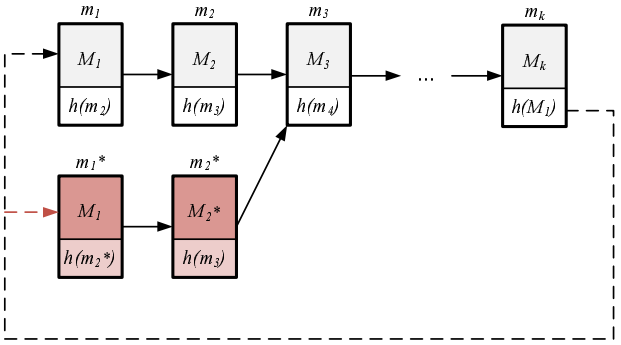


**Figure 3: We illustrate an insertion attack on the SPCC scheme, where the valid message fragment $M_2$ is replaced by the malicious fragment $M_2^*$, while the entire message is still verified by the hashes given.**

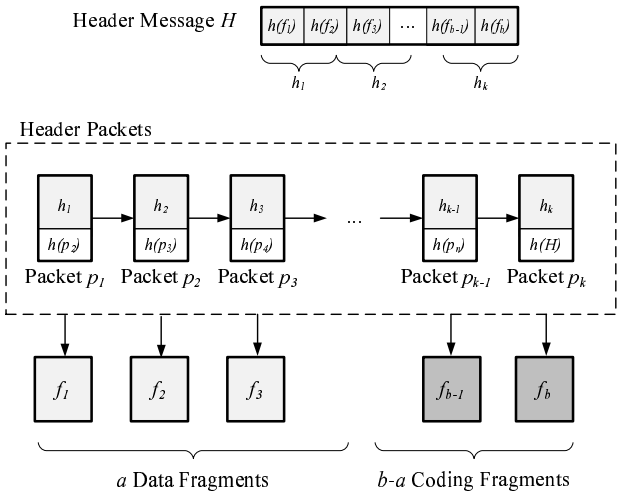


**Figure 4: The Merkleleaf scheme is illustrated. The header packets are chained together using the SPCC scheme, forming a collection of hashes of the encoded data fragments. Upon collecting the header packets, the receiver can verify any encoded data fragment. The arrows between packets and fragments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

The interested reader will note a slight difference between the Hashcluster scheme and the SPCC scheme, insofar as the hash value of the final packet. In the SPCC scheme, it is possible to insert malicious packets into the hash chain without violating the hash verification, as shown in Figure 3, thus requiring the use of secondary verification of the reconstructed message using certificates to detect the insertion. To prevent this attack, we modify the scheme so that the final packet contains a hash of the entire message instead of a hash of the first fragment. This verifies that the entire message originated from the same source, without increasing the verification complexity.

## 4.2 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.

Compared with an uncoded scheme, which for $a$ fragments required $t = aH_a$ total receptions, an $(a, b)$ erasure coded scheme would require $t = b(H_b - H_{b-a})$, which even for $b - a = 1$ significantly reduces $t$. Note that as $b \to \infty$, $t \to a$. Intuitively, decoding a $(a, \infty)$ code would require infinite computation, and equally important, a correspondingly large header message size. A consequence of using erasure coding on the data fragments is that all coded fragments need to have corresponding hashes, resulting in a larger header message than without coding, thus implying the need to fine-tune the coding rate to ensure an ideal Merkleleaf implementation.

The optimal $(a, b)$ coding rate for the data message is determined experimentally by searching through all possibilities, which are bounded by the cases of no coding, and a header message equal to the the size of the original message. Additionally, the probability $q$ that a header packet is sent, with a corresponding probability of $1-q$ for a data packet, is found through numerical methods and further developed in Section 5. An overview of the scheme is shown in Figure 4.

## 4.3 Witnesscode Scheme

We propose Witnesscode, an alternative approach based on distillation codes [10], 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 UFH, the sender will generate a set of coded fragments to send, with witnesses computed for each of them, allowing the receiver to independently verify each fragment received.

Since the sender can generate $k$ (fragment, witness) pairs at a cost of $k(k-1)$ hash operations and is unconstrained by a header message for coding length, we perform perfect erasure coding on the set of message fragments with a large number of coding fragments. This forces $t = b(H_b - H_{b-a}) \to a$ as $b \to \infty$, dramatically increasing the communication efficiency due to the elimination of redundant fragment receptions. A limiting factor in perfect erasure coding is the $O(b^2)$ computations for the receiver to decode a message. It would also be possible to use near-perfect erasure coding, which would have a complexity of $O(b)$, allowing longer codes at the expense of additional fragments needed for decoding. However, since the cost of generating

$b$ witnesses is $O(b^2)$, we keep the receiver complexity of the same order.

For the Witnesscode scheme, we use the one-way accumulator based on bilinear pairings given by the function $f$ in Section 4 of [17], resulting in a witness $w$ of size $2s$ bits. This requires $k = \lceil m/(l-2s) \rceil$ data fragments. Thus, the drawback of this scheme is a large $k$, especially when $l$ is of the same order as $s$. Additionally, the verification computation at the receiver is then $Z$ accumulator operations and at most $\lfloor Z/l \rfloor$ decoding ($O(b^2)$) and authentication operations.

# 5. ANALYSIS & SIMULATION

In order to compare the candidate schemes with the SPCC scheme in terms of communication efficiency, we use the expected number of packet receptions $t = E[V]$ as a metric, where $V$ is the number of valid packet receptions. When comparing schemes that are computationally feasible during an insertion attack, we assume that the adversary prefers to jam, simplifying our analysis to a pure jamming attack.

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. We use five different packet payload sizes $l \in \{224, 280, 336, 448, 560\}$, which includes both the data and the verification bits, and ignore the packet frame information, which can be unnecessarily specific to a given protocol. The other parameter we vary is message size, which we model from $m = 1000$ bits to $m = 4000$ bits. While we might choose to vary the triplet $(s, l, m)$, we decided to fix $s$ because there are only two degrees of freedom: the triplet $(s_0, l_0, m_0)$ yields the same results of communication efficiency as $(\alpha s_0, \alpha l_0, \alpha m_0)$. In order to compare this analysis with the simulation given in [23], their bluetooth environment had $(s = 112, l = 280, m = 2176)$.

Note that the data points on the figures comparing the various schemes have different x-coordinate values. The reason for this is that we only consider data points for each scheme where the packets are full. With the packet numbers being discrete, only specific message sizes fully saturate the packet space. Message sizes between data points are realizable, but require the $t$ value of the next largest data point, resulting in a staircase function.

## 5.1 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. Here, the worst case computation is $(Z/n)^n$ hash operations, where $Z$ is the number of received packets. Figure 5 shows $l = 280$ bits ($n = 1$ is the SPCC scheme, and $n = \infty$ is when the message is contained in a single cluster). The most significant gain is from $n = 1$ to $n = 2$. Raising $n$ further gives little gain in efficiency, but increases the complexity of the receiver by a polynomial degree at each step. Therefore, for the remainder of the simulation, we focus solely on the $n \in \{1, 2\}$ cases, with the assumption that $\frac{1}{4}Z^2$ complexity is reasonable for the receiver.

The results are determined analytically, with the principle component being the number of fragments $k$ that the message must be broken into. The difference between various $n$ for a fixed $k$ is that schemes with larger $n$ can contain a larger message, showing they lie on the same horizontal line $t = kH_k$. Note that for the $n \in \{2, 3, 4\}$ cases, their slope is rather erratic. This is due to the addition of clusters to the hash chain every $n$ increases in $k$, where the addition of a hash for that cluster leads to an increase in verification overhead.

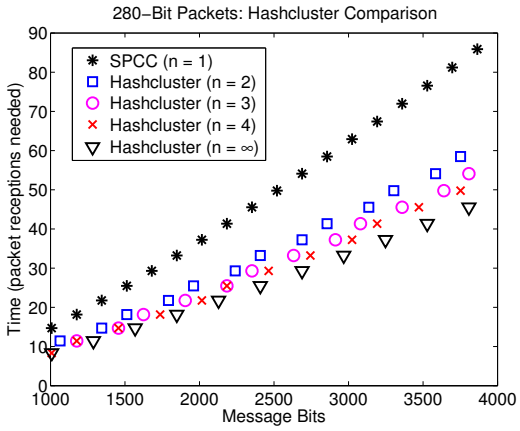


**Figure 5: The average number of packet receptions for the Hashcluster scheme is evaluated as a function of the cluster size $n$, with the SPCC scheme representing the performance baseline of $n = 1$.**

## 5.2 Merkleleaf Analysis

For the Merkleleaf scheme, there are two primary components that need to be received in order to decode, the header and the erasure coded data. Since the header packets are combined using the SPCC scheme, the time to receive them is of the same order ($kH_k$). In contrast, the coded data packets are received more efficiently. For an $(a, b)$ code with $a$ data packets and $(b - a)$ code packets, only $a$ packets need to be received in order to decode. In the absence of coding, if there is only one packet left to receive, then the probability of receiving that one (as opposed to a duplicate of another) is $1/a$, which leads to a large number of duplicates. However, in this scheme, when there is one packet left to receive, the probability of receiving that any new packet is $(b - a + 1)/b$. Thus, the number of receptions becomes $\sum_{i=b-a+1}^{b} \frac{b}{i} = b(H_b - H_{b-a}))$, which is significantly lower than the SPCC scheme.

The total time $t$ spent is then a combination of the time to receive the header and the time to receive the data. Though it is possible to analytically determine the expected time for either, the joint expected time is non-trivial, because they are dependent random processes. Specifically, the receiver must receive all of the $k$ header packets and any $a$ of the $b$ erasure coded message packets. With probability $q$, the receiver gets a header packet, uniformly from the set of $k$, and with probability $(1 - q)$, a message packet is received, likewise uniformly from the set of $b$. The result of interest is the average number of message receptions required to successfully decode both the header and data messages. This average quantity is derived by developing an appropriate two-dimensional Markov chain as follows.

The Markov chain state space $\mathcal{X}$ consists of pairs $(i, j)$ such that $0 \le i \le k$ and $0 \le j \le b$. Upon reception of a packet when in state $x = (i, j)$, the next state can be either $x$, $x_{i+} = (i+1, j)$, or $x_{j+} = (i, j+1)$, so the chain is a discrete *birth-only* chain [6]. In order to move from state $x$ to state $x_{i+}$, the received packet must be one of the $(k - i)$ header

packets that have not yet been received. Letting $X_\tau$ denote the state of the chain at (discrete) time $\tau$, the probability $p(x, x_{i+})$ of the state transition $x \to x_{i+}$ is given by

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

Similarly, to move from state $x$ to state $x_{j+}$, the received packet must be one of the $(b-j)$ message packets that have not yet been received, yielding the probability $p(x, x_{j+})$ of transition $x \to x_{j+}$ given by

$$p(x, x_{j+}) = (1-q)\left(1 - \frac{j}{b}\right). \tag{2}$$

The probability of the loop transition $x \to x$ is the remaining probability $p(x, x)$ given by

$$p(x, x) = 1 - q\left(1 - \frac{i}{k}\right) - (1-q)\left(1 - \frac{j}{b}\right). \tag{3}$$

Now that the discrete Markov chain is set up, we can approach the problem of computing the expected number of packet receptions required to decode and verify the message. This quantity is computed by defining an appropriate *hitting time* [6] for the chain. We define the *hitting set* $\mathcal{H}$ as the subset of the Markov chain state space given by

$$\mathcal{H} = \{(i, j) \in \mathcal{X} \ : \ i = k \text{ and } j \geq a\}. \tag{4}$$

We define the hitting time $T$ as

$$T = \min\{\tau \ : \ X_\tau \in \mathcal{H}\}. \tag{5}$$

To compute the expected value $t$ of the hitting time $T$, we make use of an auxiliary function $g(x)$ which gives the expected hitting time to reach $\mathcal{H}$ from the state $x \in \mathcal{X}$. The function $g(x)$ is defined recursively [6] as

$$g(x) = \begin{cases} 1 + \sum_{y \in \mathcal{X}} p(x, y) g(y) & x \notin \mathcal{H} \\ 0 & x \in \mathcal{H}. \end{cases} \tag{6}$$

Substituting the three non-zero values of $p(x, y)$ as given in (1), (2), and (3) into (6) yields the recursive formula

$$\begin{aligned} g(x) =& 1 + p(x, x_{i+}) g(x_{i+}) \\ &+ p(x, x_{j+}) g(x_{j+}) + p(x, x) g(x) \\ =& \frac{1 + p(x, x_{i+}) g(x_{i+}) + p(x, x_{j+}) g(x_{j+})}{1 - p(x, x)}. \end{aligned} \tag{7}$$

The expected hitting time $t = E[T]$ is thus computed by evaluating $g(x)$ at the initial state $x = (0, 0)$ according to the recursive formula in (7). Further investigation into a closed form solution for the function $g(x)$ as in (7) for this two-dimensional coupon collector problem is left as future work.

Given the hitting time, it is necessary to find the optimum number of packets to include in the header and the coding, of which there are many. For instance, consider $l = 336$ bits. This leaves space for two hash values of data packets in each header packet. Thus, if there are $k$ header packets, the number of coded packets is $b = 2k$. Let's suppose that we are trying to find the optimal configuration for transmitting $m = 2016$ bits, which results in $k = 6$ data packets. (If we were using the original scheme, it would take $k = 9$ packets) The possible triplets $(k, a, b)$, where $b$ is the erasure coded message, is {(4, 6, 8), (5, 6, 10), (6, 6, 12), (7, 6, 14), (8, 6, 16)}. The triplet (3, 6, 6) would be equivalent to the SPCC scheme, because all $3 + 6 = 9$ packets would need to be received. Also, (9, 18, 18) would be pointless, because the header contains the same amount of information as the coded data, so there is no reason for the data packets, but the data packet hashes should be replaced with the data itself. In order to determine the optimal configurations, for each data point we analyzed all possible triplets and chose the most efficient. Furthermore, we determined the optimal probability $q$ of receiving a header packet using numerical methods. For this example, the optimal solution was $(k, a, b) = (4, 6, 8)$ and $q = 0.461$. It was assumed that any fractional leftover information in the header (less than $s$, so insufficient to contain a hash) was discarded. This occurrence only took place in the 280-bit packets, where the payload length was $2.5s$.

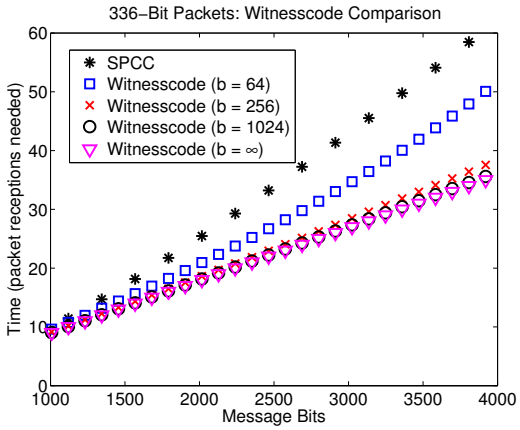


**Figure 6: The average number of packet receptions for the Witnesscode scheme is evaluated for various coding values $b$, with the SPCC scheme included for reference. The case of $b = \infty$ represents the optimal coding case with infinite computational overhead.**

## 5.3 Witnesscode Analysis

For the Witnesscode scheme, a similar analysis can be done. The coding parameter in this scheme should be adjusted so that the computational complexity at the sender and receiver are still feasible. At the sender, it takes $b(b-1)$ two-to-one ECC hash operations, where $b$ is the total number of packets (data and coding). At the receiver, the Reed-Solomon coding takes $O(b^2)$ operations. To determine an adequate tradeoff between computation and communication, in Figure 6, we show the results of several values of $b$ for the accumulator scheme with $s = 112$, $l = 336$. 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 $b = 1024$ case, because it has a reasonable computational complexity for practical purposes and is extremely close to the optimal in terms of communication efficiency.

## 5.4 Scheme Comparison

Next, we show the bulk of the simulation results, comparing the SPCC, Hashcluster ($n = 2$), Merkleleaf, and Wit-

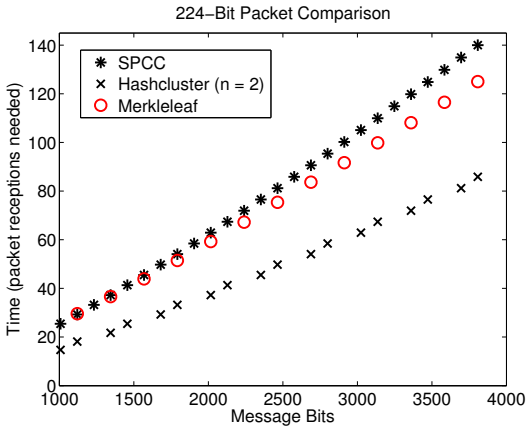


**Figure 7: The average number of packet receptions for the SPCC, Hashcluster, and Merkleleaf schemes are compared for 224-bit payloads and 112-bit hashes.**

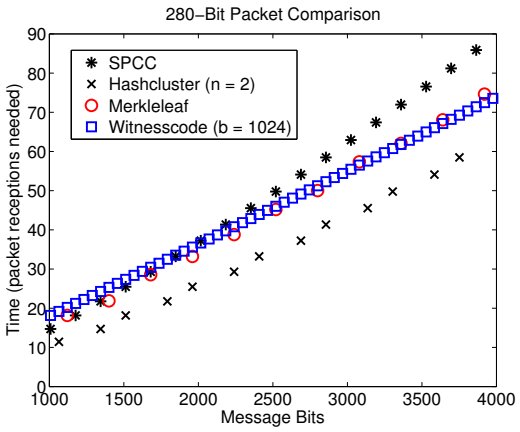


**Figure 8: The average number of packet receptions for the SPCC, Hashcluster, Merkleleaf, and Witnesscode schemes are compared for 280-bit payloads and 112-bit hashes.**

nesscode ($b = 1024$) schemes for $s = 112$, $m \in [1000, 4000]$, and $l \in \{224, 280, 336, 448, 560\}$. These are shown in Figures 7, 8, 9, 10, and 11, respectively. Note that the Witnesscode scheme is not shown for the 224-bit case. This is because in this case the witness saturates the payload of the packet, reducing the data to zero.

To describe the benefits to each scheme, we begin with SPCC. First, as is clear from the preceding figures, when the message is small (close to 1000 bits), the SPCC scheme has an efficiency roughly equivalent to the other three schemes. Since it is the simplest to implement, and also 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 of packets is likewise small, thus reducing the probability of receiving additional redundant packets. For this reason, decreasing the message size below 1000 bits will only further improve the SPCC scheme, making it ideal for short messages.

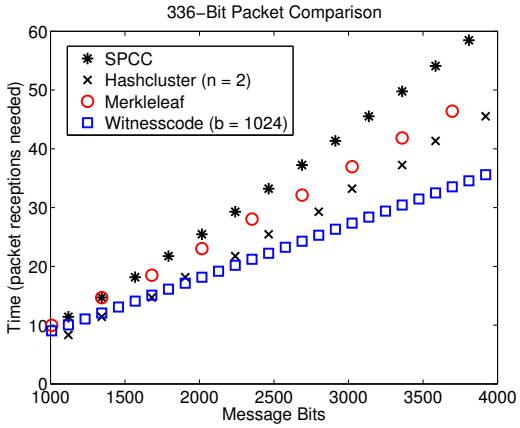


**Figure 9: The average number of packet receptions for the SPCC, Hashcluster, Merkleleaf, and Witnesscode schemes are compared for 336-bit payloads and 112-bit hashes.**

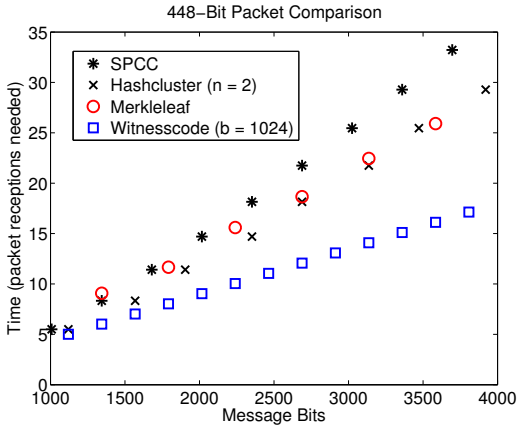


**Figure 10: The average number of packet receptions for the SPCC, Hashcluster, Merkleleaf, and Witnesscode schemes are compared for 448-bit payloads and 112-bit hashes.**

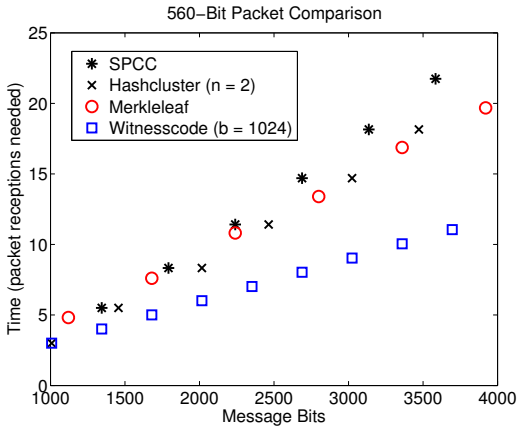


**Figure 11: The average number of packet receptions for the SPCC, Hashcluster, Merkleleaf, and Witnesscode schemes are compared for 560-bit payloads and 112-bit hashes.**

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, resulting in $\lceil k/n \rceil$ hashes. 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 hash cluster scheme performs especially well for small packet sizes, where using this scheme will free up a greater fraction of the payload for data. For larger packet sizes, its effect is less evident, as an insignificant portion of the packets are used to store hashes.

The Merkleleaf scheme, while its gains are modest and is suboptimal in terms of communication efficiency, has nearly the same computational complexity of the SPCC scheme, and overtakes it in terms of efficiency for message sizes larger than about 2000 bits. For larger message sizes, it can take the communication down to 80-95% of the SPCC. The only additional computation of note is the Reed-Solomon coding on the data packets. However, since the code length $b$ is at most 20 for all considered values of $m$ and $l$, the $O(b^2)$ decoding operation is insignificant. When it is computationally infeasible to use the Hashcluster or Witnesscode schemes, this becomes the optimal approach.

Finally, the Witnesscode scheme has its greatest benefit for large packet sizes and message lengths. For the former case, the witness being twice the length of the typical hash has a lessened effect. For the latter case, having a large message implies a large number of packets, which was where the inefficiency in the first three schemes was located. In this case, however, the coding gain 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. While the first three schemes' times $t$ increase as $O(k \log_2 k)$, the accumulator increases as $O(k)$ for $b \gg a$, albeit with a larger constant.

# 6. CONCLUSION

In this paper, we addressed the problem of allowing authorized users to efficiently exchange key establishment messages in the presence of jamming and message insertion attacks. This problem was first introduced and studied in [23], and the method we refer to as SPCC was proposed. The SPCC scheme was introduced to provide the security of the packet verification, while communication efficiency was not addressed. The authors have recently extended this work to broadcast communication [19]. In this work, we jointly consider the security of packet verification and the total time required for key establishment. We proposed three candidate protocols for packet verification in conjunction with the UFH communication model, leading to a reduction in the total time required for key establishment in comparison to the SPCC scheme. We first presented the Hashcluster scheme which reduces the overhead included in the hash chain used for packet verification. Next, we presented the Merkleleaf scheme which uses erasure coding to reduce the average number of packet receptions required without increasing the computational complexity. We then presented the Witnesscode scheme which uses one-way accumulators to individually verify packets and reduce the level of redundancy among received packets. We demonstrated the improvements in communication efficiency through analysis and simulation of the three proposed schemes and compared to the existing SPCC scheme. We note that the communication efficiency of packet verification can be further improved by combining the three proposed schemes. For example, by creating a hybrid scheme between the Hashcluster and Merkleleaf schemes, using the former to verify the header packets in the latter, the average number of packet receptions can be further decreased. Future work includes further generalization of the Merkleleaf scheme using hash trees or more general hash graphs [7] and the extension of erasure coding to rateless erasure codes [15] aided by the use of homomorphic encryption [9].

# 7. REFERENCES


[1] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

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

[3] 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.

[4] 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.

[5] 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.

[6] R. Durrett. *Essentials of Stochastic Processes*. Springer-Verlag, Inc., 1999.

[7] P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss. In *Proc. of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pages 13–22, Feb. 2001.

[8] 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.

[9] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. *Advances in Cryptology – EUROCRYPT 2000*, pages 539–556, 2000.

[10] 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.

[11] 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.

[12] 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.

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

[14] M. Luby. LT codes. In *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '02)*, pages 150–159, May 1997.

[15] P. Maymounkov. Online codes. *NYU, Tech. Rep. 2002-833*, Nov. 2002.

[16] R. Merkle. Protocols for public key cryptosystems. In *Proc. 1980 IEEE Symposium on Security and Privacy*, pages 150–159, Apr. 1980.

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

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

[19] C. Pöpper, M. Strasser, and S. Čapkun. Jamming-resistant broadcast communication without shared keys. Technical Report 609, ETH Zurich, Sept. 2008.

[20] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2 edition, 2001.

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

[22] A. Shokrollahi. Raptor codes. *IEEE/ACM Transactions on Networking (TON)*, 14:2551–2567, 2006.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.