Creating Secrets out of Erasures

Katerina Argyraki[1] [*], Suhas Diggavi[2] [†], Melissa Duarte[1],
Christina Fragouli[1] [‡], Marios Gatzianas[1], Panagiotis Kostopoulos[1]

[1]EPFL, Switzerland [2]UCLA, USA

ABSTRACT

Current security systems often rely on the adversary's computational limitations. Wireless networks offer the opportunity for a different, complementary kind of security, which relies on the adversary's limited network presence (i.e., that the adversary cannot be located at many different points in the network at the same time). We present a system that leverages this opportunity to enable n wireless nodes to create a shared secret $\mathcal{S}$, in a way that an eavesdropper, Eve, obtains very little information on $\mathcal{S}$. Our system consists of two steps: (1) The nodes transmit packets following a special pattern, such that Eve learns very little about a given fraction of the transmitted packets. This is achieved through a combination of beam forming (from many different sources) and wiretap codes. (2) The nodes participate in a protocol that reshuffles the information known to each node, such that the nodes end up sharing a secret that Eve knows very little about. Our protocol is easily implementable in existing wireless devices and scales well with the number of nodes; these properties are achieved through a combination of public feedback, broadcasting, and network coding. We evaluate our system through a 5-node testbed. We demonstrate that a group of wireless nodes can generate thousands of new shared secret bits per second, with their secrecy being independent of the adversary's computational capabilities.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; H.1.1 [**Information Systems**]: Systems and Information Theory—*Information theory*

Keywords

Physical-layer security; Group secret agreement

1. INTRODUCTION

To perform secret agreement in the presence of an adversary, current security systems rely on the adversary's computational limitations. Consider the scenario where a principal Alice wants to agree on a secret with a principal Bob over a communication channel, while an adversary Eve is eavesdropping on the same channel. Today, we solve this problem through algorithms like Diffie-Hellman [10] or RSA [27], which rely on the assumption that Eve cannot perform certain operations, e.g., large integer factorization, in useful time.

Wireless networks offer the opportunity for a different kind of secret agreement, which relies not on the adversary's computational limitations, but on her limited network presence (i.e., the fact that the adversary cannot be located at many different points in the network at the same time). Consider the scenario where Alice, Bob, and Eve are standing in a noisy room; when Alice whispers something, Bob hears some part of it, Eve hears another part, but, as long as Bob and Eve are not standing right next to each other, they are highly unlikely to hear exactly the same thing. Information theory tells us that Alice and Bob can agree on a secret that Eve knows nothing about, as long as they know how much information Eve has heard (e.g., they know that Eve heard half of Alice's worlds, even though they do not know which ones) [30, 21]. This result holds, even when the Alice/Bob channel is worse than the Alice/Eve channel, as long as Eve does not always hear every single bit of information that Bob does [30, 21]. So, it is theoretically possible for Alice and Bob to agree on a secret that Eve knows nothing about, even when Eve can hear Alice more clearly than Bob can.

This result has significant practical implications: It opens up the possibility of wireless nodes continuously sharing fresh secrets, simply by communicating with each other in a noisy environment. Such a "secret stream" can be used to periodically refresh the nodes' encryption and/or authentication keys [31]. This is important in light of the fact that security is most often compromised through stolen or weak keys, not vulnerabilities in the underlying algorithms [31].

But there is a second, equally important (if more academic and longer-term) reason to care for this line of work: As powerful governments and corporations amass computational power, secret agreement that makes no assumption about the adversary's computational capabilities will become a relevant problem. We do not expect this to happen tomorrow, and we are by no means advocating the replacement of existing cryptosystems that rely on the adversary's computational limitations, like RSA. But we do think that there is value in researchers starting to think about alternatives,

[*] Supported by ArmaSuisse.

[†] Supported in part by NSF award 1136174 and MURI award AFOSR FA9550-09-064.

[‡] Supported by ArmaSuisse and ERC Starting Grant ERC-2009-StG-240317.

MobiCom'13, September 30–October 4, Miami, FL, USA.
Copyright 2013 ACM 978-1-4503-1999-7/13/09 ...$15.00.
http://dx.doi.org/10.1145/2500423.2500440.

even if these are bound to suffer, at first, from vulnerabilities and deployment hurdles.

Our goal then is to design and build a system that enables n wireless nodes (we will call them "terminals") to agree on a secret $\mathcal{S}$, in a way that an eavesdropper, Eve, obtains very little information on $\mathcal{S}$. We focus on groups (as opposed to pairs) of terminals, because this scenario has received less attention by the research community, even though it is becoming increasingly relevant: Internet users are increasingly relying on their mobile phones to communicate in groups, e.g., participate in social networks and online games. There is also evidence that they like to consume online content (in particular, videos) in groups [5], and we are starting to see commercial tools that explicitly target groups of mobile users accessing the same content [4, 2].

Our system does not assume anything about Eve's computational capabilities, but it assumes limits on her network presence, e.g., that she possesses no more than a certain number of antennas. Information theory tells us that this is theoretically possible for $n = 2$ terminals, but not how to do it. In particular, the existing theoretical results (upper and lower bounds on the secrecy rate achievable between two nodes given idealized channel conditions) rely on combinatorial constructions that are not implementable in practice. More recently, researchers have started to propose implementable solutions for $n = 2$ terminals. The first proposals assumed channel reciprocity between the two nodes and required a dynamic environment [13, 7, 32, 8, 17], or else risked compromising their security. A more recent one avoids these pitfalls but requires physical-layer changes and is specifically designed for OFDM [12]. To the best of our knowledge, there exists no solution (for 2 or more terminals) that does not require physical-layer changes, has been deployed on a real testbed, and has been experimentally shown to generate high-quality secrets in the presence of an adversary.

We present such a solution that consists of two phases: In the first phase, the terminals transmit packets following a special pattern, such that Eve learns very little about a given fraction of the transmitted packets; this is achieved through a combination of beamforming (from many different sources) and wiretap codes. In the second phase, the terminals participate in a protocol that reshuffles the information known to each terminal, such that the terminals end up sharing a secret that Eve knows very little about; this is achieved through a combination of public feedback, broadcasting, and network coding.

We evaluate our system on a small testbed of 5 WARP nodes, 4 of them playing the role of the terminals and one of them playing the role of the eavesdropper Eve. We show that, for a variety of node placements (all the placements we tried), the terminals agree on thousands of new secret bits per second. We verify that these bits are perfectly secret (Eve obtains no information about them) by measuring the mutual information between the signal that reaches Eve's antenna and the signals that reach the antennas of the terminals. In other words, we ensure that Eve cannot learn anything about the shared secrets, even if she has access to corrupted packets that get discarded by the lower layers of her device.

2. SETUP AND OVERVIEW

In this section, we state the input and output of our system (Section 2.1), our adversary model (Sections 2.2 and 2.3), the basic elements of our approach (Section 2.4), and its limitations (Section 2.5).

2.1 Goal

We consider n wireless nodes, $T_0, \ldots, T_{n-1}$, connected to the same broadcast channel; we will refer to these nodes as *terminals*; sometimes we will refer to terminals T_0, T_1, and T_2 respectively as Alice, Bob, and Calvin. We also consider an adversary node, Eve, connected to the same broadcast channel as the terminals.

The terminals can communicate with each other in two ways: (1) When we say that terminal T_i *transmits* a packet, we mean that it broadcasts the packet once. (2) When we say that terminal T_i *reliably broadcasts* a packet, we mean that it ensures that all other terminals T_i receive it, e.g., through acknowledgments and retransmissions. To be conservative, we assume that Eve receives all reliably broadcast packets.

We design a system that enables the n terminals to create a shared secret $\mathcal{S}$, in a way that Eve obtains very little information on $\mathcal{S}$. We will make no assumptions about Eve's computational capabilities, but we will make assumptions about her network presence. The input to our system is: (a) the physical locations of the terminals and certain properties of the terminals' hardware, in particular, the shape of their primary antenna lobe and the SNR range of a receiver located inside and outside the primary lobe; (b) an assumption about Eve's hardware, in particular, the minimum noise experienced by Eve's receiver relative to the noise experienced by the terminals' receivers.

2.2 Adversary Capabilities

The rate at which the terminals can agree on new secret bits depends on their physical locations and hardware, as well as Eve's physical location and hardware. Since we cannot know where Eve is or what kind of hardware she has, we design our system to be configurable for different adversaries. In other words, the terminals need to change the parameters of the system depending on how strong an adversary they want to be secure against. Intuitively, the stronger the adversary, the lower the rate at which the terminals can agree on new secret bits.

As mentioned above, our system takes as input certain properties of the terminals' hardware and an assumption about Eve's hardware. Based on that, it estimates: (a) what is the maximum rate at which the terminals can create new shared secret bits; (b) what is the minimum physical distance δ between any terminal and Eve for which the system is secure (i.e., Eve knows very little about the created shared secret). To be conservative, the terminals can (and should) configure for the case where Eve has a better receiver than they do. For example, a group of terminals may configure the system to be secure as long as Eve's receiver experiences at least a quarter of the noise experienced by any terminal's receiver; the system may estimate that, given this noise ratio and the terminals' hardware, the terminals can create up to 10 secret Kbps, as long as Eve is located, say, 3 m from any terminal.

We assume that Eve has access to all the layers of her device, including the physical-layer signal that reaches her receiver. So, when Eve tries to reconstruct the terminals' communication (in order to guess the secret $\mathcal{S}$), we assume that she can exploit all the data that reaches her, including data with errors that may get dropped by the higher layers of her communication device.

2.3 Adversary Behavior

When we say that Eve is "passive," we mean that she does not perform any transmissions, she only processes the traffic she overhears from the terminals.

When we say that Eve is "active," we mean that, on top of processing the traffic she overhears from the terminals, she transmits her own traffic. The goal of Eve's transmissions may be (1) to jam the wireless channels to prevent secret agreement from occurring in the first place, or (2) to impersonate a terminal (e.g., pretend to Bob that she is Alice).

Attack type	Outcome	Extra mechanism
Passive	completes	–
Jamming	aborts	–
Impersonation	completes	bootstrap auth.

Table 1: Outcome of our system under attack.

Table 1 summarizes what our system is meant to do in each case, and whether it needs any extra mechanism to do it (on top of what we describe in this paper).

In case of a passive attack, secret agreement should complete successfully. This should not require any extra knowledge or mechanism, in particular, the terminals do not need to share any bootstrap information to start the agreement. This is the main scenario that drives our design.

Our current design does not handle jamming attacks, in the sense that the terminals abort their secret agreement when they cannot establish among them communication channels of sufficient quality. At this stage of the work, we consider it acceptable that the terminals realize that they cannot agree on a secret and abort (as opposed to agreeing on a secret that is compromised by Eve).

In case of an impersonation attack, secret agreement should complete successfully, with the help of a bootstrap authentication mechanism. Once the terminals have agreed on a secret $\mathcal{S}$ using our system, they can use part of $\mathcal{S}$ to form authentication codes and authenticate each other (during subsequent uses of the system). The first time they use the system, however, they need to rely on some external authentication mechanism. For example, the terminals may share an initial secret σ (much smaller than $\mathcal{S}$) and use it to form initial authentication codes. This is fundamentally unavoidable: the very first time Alice sends packets to Bob, she must have a way to prove that it is she (not Eve) sending the packets.

One may ask: if the terminals need to already share an initial secret σ (to authenticate each other when they first use our system), why would they use our system at all? The answer is that our system enables the terminals to continuously create new secrets that are completely independent from σ (and from each other), without using any out-of-band channel. So, even if the adversary at some point cracks σ, that will not give her any information on the secrets created with our system. Hence, as long as our system works as it is supposed to, the only way an adversary can compromise it is to crack σ, cancel out a terminal's transmissions, impersonate that terminal, and participate in secret agreement, *while the terminals are using our system for the first time*, i.e., before they have agreed on their first secret $\mathcal{S}$.

2.4 Approach

The insight we use from information theory is that, when Alice transmits, it is possible for Alice and Bob to agree on a secret, as long as Eve does not receive exactly the same information as Bob.

Our system then consists of two phases: in the first phase, the terminals exchange random packets using a special pattern, which ensures that Eve misses some fraction of the exchanged packets; in the second phase, the terminals participate in a polynomial-time protocol, which enables them to agree on a common secret $\mathcal{S}$. We would like to emphasize two points: First, we do not need to know or control which packets exactly Eve misses, it is sufficient to know that she has missed *some fraction* of the packets exchanged by the terminals in phase 1. Second, we do not assume the existence of natural erasure channels: when a terminal transmits a frame, Eve may receive only a part of that frame correctly and still extract useful information from it. This is why we use a special transmission pattern in phase 1—to ensure that, despite receiving partial frames, Eve will still completely miss some fraction of the packets exchanged by the terminals.

We now outline each phase:

First, we create channel variability, i.e., ensure that no two nodes (including Eve and any terminal) hear exactly the same information. To achieve this, in phase 1, the terminals take turns in transmitting random packets while beamforming and rotating their beams. Beamforming enables a transmitter to concentrate its power along a narrow beam, such that receivers within the beam receive a significantly stronger signal than those outside; rotating the beam keeps changing the sets of receivers that are in and out of the beam. Hence, by having the terminals take turns in beamforming while rotating, we ensure that no two nodes hear all transmissions with the same signal strength.

Next, we turn channel variability into packet erasures, i.e., ensure that any node (including Eve) completely misses some fraction of the transmitted packets. To achieve this, when the terminals transmit in phase 1, they use wiretap codes [18]. These create artificial erasure channels at the cost of extra bandwidth: they augment the packets generated by the terminals with redundant information, enough to ensure that any receiver with a signal-to-noise ratio (SNR) below a given threshold can extract zero information about any transmitted packet.

So, at the end of phase 1, the terminals share (know the contents of) some packets, while Eve has missed (knows nothing about the contents of) some fraction of these packets; the terminals know a lower bound of how many packets Eve has missed, but not which ones. This lower bound is computed based on the input to our system (the terminals' hardware properties and the assumptions about Eve's hardware properties).

In phase 2, the terminals agree on a common secret $\mathcal{S}$. To achieve this, they mostly perform simple linear combinations of the packets they share at the end of phase 1. For example, suppose we have $n = 2$ terminals, Alice and Bob; at the end of phase 1, Alice and Bob share 3 packets, x_1, x_2 and x_3, while Eve has missed two of these packets, x_1 and x_2. If Alice and Bob know that Eve misses two of their shared packets (but not which two), they can create a shared secret that Eve knows nothing about, by concatenating two linear combinations of their shared packets, e.g., $\langle x_1 \oplus x_2, x_2 \oplus x_3 \rangle$ (where $\oplus$ denotes the bit-wise XOR operation over the payloads of the corresponding packets). The actual protocol is more sophisticated, as it must ensure that Eve cannot guess $\mathcal{S}$ no matter which particular packets she has missed, and it must scale well with the number of terminals.

The size of the resulting common secret $\mathcal{S}$ depends on the smallest number of packets shared by any terminal pair and missed by Eve at the end of phase 1. For instance, in the above example, Eve misses 2 of the packets shared by Alice and Bob; hence, the secret $\mathcal{S}$ should have at most the size of 2 packets. If the terminals created a bigger secret, then Eve would be able to guess something about it. So, our protocol is secure, as long as we can lower-bound the number of packets missed by Eve in phase 1. We can obtain such a lower bound thanks to the combination of beamforming and wiretap codes.

2.5 Limitations

We have not considered yet the case where Eve has multiple receivers. Our current system takes as input the assumed level of noise at Eve's receiver; to deal with a multi-receiver Eve, it should also take as input the assumed number of receivers R that Eve controls. We currently upper-bound the amount of information that Eve overhears in phase 1, by assuming that she is located in the worst possible position, e.g., in the middle of all the terminals (Section 4); to deal with a multi-receiver Eve, we would have to be more conservative, i.e., assume that Eve's R receivers are located in the worst possible R positions with respect to the terminals. The next question that we want to answer is how many receivers R Eve must control in order to make the creation of a secret between n terminals impossible.

We have experimented only with the simplest wiretap codes (two-layer codes). However, we have observed evidence that using more sophisticated wiretap codes (with more than two layers) would significantly improve the performance of our system.

Our experimental evaluation is only a first step, and we need to improve it in at least three ways:

(a) The performance of our system depends on the SNR range of a receiver located inside and outside the transmitter's primary lobe (Section 4). We need to study how sensitive these SNR ranges are to the particular environment (e.g., indoors versus outdoors, node hardware, antenna specifications) and make our system robust to environmental differences. So far, we have experimented only with one environment (outdoors, WARP wireless nodes [3], and flat-panel directional antennas [1]).

(b) The performance of our system also depends on the shape of the terminals' beam (Section 4). The antennas we are currently using have relatively poor directionality (the primary lobe opens at a 40 degree angle), which imposes a minimum physical distance of 3 meters between any terminal and Eve. We need to experiment with higher-quality antennas that can focus their beams at a smaller angle, so that we decrease this minimum physical distance to less than 1 meter.

(c) Our system relies on properties of simple wiretap codes (with two layers). We still need to experimentally validate practical implementations of such codes.

3. SECRET AGREEMENT

In this section, we describe phase 2, i.e., our secret-agreement protocol: the input it expects from phase 1 (Section 3.1), its two algorithms (Sections 3.2 and 3.3), and its main properties (§3.4).

At a high level: Alice first agrees on a pairwise secret $\mathcal{S}_i$ with each terminal T_i, such that Eve knows nothing about $\mathcal{S}_i$. Once Alice has created a pairwise secret with each terminal, she could use these pairwise secrets to communicate a group secret to all the terminals. Perhaps counter-intuitively, there exists a significantly more efficient way to agree on a group secret: Alice reliably broadcasts certain carefully chosen information, which does not increase the amount of secret information shared by Alice and each terminal, but "redistributes" it, such that all terminals share the same group secret $\mathcal{S}$.

3.1 Input

At the end of phase 1, the terminals have transmitted some number of packets (we will call them *x-packets*). Each terminal, as well as Eve, has received and knows the contents of some fraction of these x-packets.

The input to the secret-agreement protocol consists of:

- The identities of the x-packets known to each terminal.
- A lower bound on the number of x-packets shared by each pair of terminals and missed by Eve.
- A lower bound on the number of x-packets shared by at least two terminals and missed by Eve.

Obtaining this input is the topic of the next section.

3.2 Pairwise-Secret Algorithm

The terminals take turns playing the role of Alice and running the following algorithm:

1. Alice constructs linear combinations of the x-packets that she knows (we will call them *y-packets*), using a well-defined construction (see Appendix, Section 8.2). These y-packets satisfy two constraints:

   (a) Their total number is M.

   (b) M_i of them are linear combinations of the x-packets that Alice shares with terminal T_i.

   Alice reliably broadcasts the identities of the x-packets she used to create each y-packet.

2. Each terminal T_i (other than Alice) reconstructs the contents of M_i y-packets.

At this point, Alice and terminal T_i share M_i y-packets. The pairwise secret $\mathcal{S}_i$ is their concatenation.

For example, suppose we have $n = 2$ terminals, Alice and Bob. At the end of phase 1, Alice and Bob share x-packets x_1, x_2, x_3, x_4, x_5, of which Eve knows x_1, x_2, x_3. The pairwise-secret algorithm works as follows: In step 1, Alice constructs $M_1 = 2$ y-packets that are linear combinations of the x-packets shared with Bob: $y_1 = x_1 \oplus x_3 \oplus x_5$ and $y_2 = x_2 \oplus x_4$. Then, Alice reliably broadcasts the identities of the x-packets she used to construct the y-packets, but not their contents. In step 2, Bob uses this information to reconstruct the contents of the y-packets. Eve overhears Alice's reliable broadcast, but she cannot reconstruct the contents of either y_1 or y_2, because she does not know the contents of x_4 or x_5. At this point, Alice and Bob have agreed on pairwise secret $\langle y_1, y_2 \rangle$, and Eve knows nothing about it.

It is important that Alice construct the y-packets using a particular construction, because not any linear combinations of x-packets will do. The y-packets that we use above happen to work for the particular example, but our protocol does not really construct so simple linear combinations, as they may leak information to Eve. For instance, suppose Alice constructed the y-packets as follows: $y_1' = x_1 \oplus x_2 \oplus x_3$ and $y_2' = x_4 \oplus x_5$. In this case, Eve would be able to reconstruct y_1', hence recover half of the pairwise secret.

In this example, Eve knows nothing about the pairwise secret $\langle y_1, y_2 \rangle$, because it consists of 2 linear combinations of the x-packets, which is exactly the number of x-packets shared by Alice and Bob and missed by Eve at the end of phase 1. So, to create a perfect pairwise secret with terminal T_i, Alice needs to know a lower bound on the number of x-packets shared with T_i and missed by Eve, and she must set M_i (the size of the pairwise secret) to this value.

3.3 Group-Secret Algorithm

Next, the terminals take turns playing Alice and running a second algorithm. We first present a simplified version of it:

1. Alice constructs $M - \min\{M_1, M_2, \ldots M_{n-1}\}$ linear combinations of the y-packets (we will call them *z-packets*), using a well-defined construction (see appendix 8.2). She reliably broadcasts both the contents of each z-packet and the identities of the y-packets used to construct each z-packet.

2. Each terminal T_i reconstructs the $(M - M_i)$ y-packets it is missing by combining any of the $(M - M_i)$ z-packets with the M_i y-packets it reconstructed in step 2 of the pairwise algorithm.

3. Alice constructs $L = \min\{M_1, M_2, \dots M_{n-1}\}$ linear combinations of the y-packets (we will call them *s-packets*), using a well-defined construction (see appendix 8.2). She reliably broadcasts the identities of the y-packets that she used to create each s-packet.

4. Each terminal is now able to reconstruct all the s-packets, because it has all the y-packets.

At this point, all terminals share the same set of L s-packets. The group secret $\mathcal{S}$ is their concatenation.

For example, suppose we have three terminals, Alice, Bob, and Calvin. After running the pairwise algorithm, Alice has constructed $M = 3$ y-packets. Of these, she shares $M_1 = 2$ with Bob (y_1 and y_2) and $M_2 = 2$ with Calvin (y_1 and y_3). Eve knows nothing about any of the y-packets.

First, we reach a point where all terminals share all the y-packets: In step 1, Alice constructs $M - \min\{M_1, M_2\} = 1$ linear combination of the y-packets, $y_2 \oplus y_3$, and reliably broadcasts its contents. In step 2, each of Bob and Calvin uses this information to reconstruct the y-packet that he is missing. At this point, all terminals share y_1, y_2, y_3, while Eve knows the value of $y_2 \oplus y_3$.

Next, the terminals condense the y-packets into a group secret: In step 3, Alice constructs $L = \min\{M_1, M_2\} = 2$ s-packets that are linear combinations of the y-packets: $s_1 = y_1 \oplus y_2 \oplus y_3$ and $s_2 = y_1 \oplus y_2$. Then, Alice reliably broadcasts the identities of the y-packets she used to create the s-packets, but not their contents. In step 4, Bob and Calvin use this information to reconstruct the contents of the s-packets. Eve overhears Alice's reliable broadcast, but she cannot reconstruct the contents of s_1 or s_2, because she does not know the contents of y_1 or y_2. At this point, Alice, Bob, and Calvin have agreed on group secret $\langle s_1, s_2 \rangle$, and Eve knows nothing about it.

As with the y-packets, it is important that Alice construct the z-packets and s-packets using specific constructions, otherwise they may leak information to Eve.

The size L of the final group secret is the size of the smallest pairwise secret ($\min M_i$). So, the group-secret algorithm does not increase the amount of secret information shared by Alice and any terminal, but it "redistributes" this information, such that all terminals share the same group secret.

What we presented above is a simplified version of the group secret algorithm: according to our description, each terminal plays the role of Alice and creates a separate group secret with the other terminals; in the actual algorithm, all the terminals together act as one "distributed Alice" and create a single group secret (see Appendix, Section 8.3).

3.4 Properties

Our secret-agreement protocol starts from the assumption that each terminal pair shares some x-packets that Eve has missed. It builds on this assumption to create, first pairwise secrets between terminal pairs, then a common group secret $\mathcal{S}$ among all terminals.

The protocol is secure, in the sense that Eve knows nothing about the group secret $\mathcal{S}$ (or the pairwise secrets), as long as the input to the protocol (the lower bounds on what Eve knows) are correct.

Equivalently, to ensure that Eve knows nothing about the group and pairwise secrets, we need to restrict their sizes: The size of each pairwise secret $\mathcal{S}_i$ must be no more than the number of x-packets M_i shared by Alice and T_i and missed by Eve at the end of phase 1. The group secret $\mathcal{S}$ must be no bigger than the smallest pairwise secret $\mathcal{S}_i$. So, the security of the protocol depends on our ability to lower-bound the number of x-packets shared by different terminal pairs and missed by Eve at the end of phase 1.

The intuition behind the security of the protocol is the following: If Alice and Bob share N packets, of which Eve misses M (but we do not know which ones), Alice and Bob can create exactly M linear combinations of their N shared packets that are perfectly secret from Eve. This is always the case, as long as N is above a given threshold.

A practical question is how to pick linear combinations that are guaranteed to be unknown to Eve and linearly independent, no matter which particular M packets she has missed; we do so with the help of Maximum Distance Separable (MDS) codes. MDS codes (which were created for entirely different purposes) have a convenient property: any M columns of the $M \times N$ generator matrix of an $[N, M, N - M + 1]$ MDS code are linearly independent. Thus, if we use this generation matrix to create M linear combinations of N unknowns, and then set to zero any $N - M$ of these unknowns (i.e. remove them from the equations because Eve knows them), the M equations (of the remaining M unknowns) are still linearly independent.

The complexity of the protocol is polynomial in the size of the secret and comes mainly from the encoding operations that Alice has to perform. We note that there exist efficient implementations, and similar-complexity encoding operations are often performed by wireless systems today.

4. CREATING ERASURE CHANNELS

In this section, we describe phase 1, i.e., how the terminals create erasure channels between themselves and Eve (Section 4.1), and how they estimate the input needed for the next phase (Section 4.2).

At a high level: The terminals take turns in transmitting random packets while beamforming and rotating their beams. When they are done, they exchange information on which packets each of them received correctly, and they estimate conservative lower bounds on the number of packets missed by Eve (the input to phase 2).

4.1 Wiretap Coding and Beamforming

Our secret-agreement protocol assumes erasure channels between the terminals and Eve: when a terminal transmits, Eve either receives a packet correctly or not at all. This assumption is fundamental to the correctness of the protocol (if it does not hold, Eve may collect significant information about the secret).

Since erasure channels do not occur naturally, the terminals use two-layer wiretap codes to emulate them: Such a code is characterized by two parameters, SNR_1 and SNR_2. When Alice transmits using such a code, it is guaranteed that Bob will receive all her packets correctly, while Eve will receive zero information about any of them, as long as Bob hears Alice with SNR above SNR_1 and Eve hears Alice with SNR below SNR_2. Hence, as long as there is a sufficient gap between Bob's and Eve's SNRs, the wiretap code ensures that Bob will receive everything and Eve will receive nothing. In this sense, wiretap coding translates SNR differences into erasures (for a brief background on wiretap codes, please see Appendix, Section 8.1).

Our main idea is the following: Provided that Alice, Bob and Eve are physically separated by some minimum distance, when Alice transmits, as she rotates her beam, there will exist some time interval during which Bob will be inside Alice's primary lobe, while Eve will be outside (unless Alice, Bob, and Eve are located on a single line). Suppose we could guarantee that, during this time interval, Bob hears Alice with SNR above SNR_1, while Eve hears Alice with SNR below SNR_2. Then, we could also guarantee that Bob will receive correctly all the packets transmitted by Alice during this time interval, while Eve will receive zero information about any of them. To achieve this, we need to set SNR_1 to the minimum

SNR that Bob may experience when he is inside Alice's primary lobe and SNR_2 to the maximum SNR that Eve may experience when she is outside Alice's primary lobe (while also satisfying the minimum acceptable physical distance from Alice and Bob). In short, Alice transmits with a wiretap code such that: when Bob is inside her primary lobe and Eve outside, Bob receives everything and Eve nothing.

One challenge here is how to practically estimate the two thresholds, SNR_1 and SNR_2. Fortunately, we do not need to be accurate, merely conservative. One approach is to use the specifications of the terminals' antennas and plug them into an analytical model, e.g., the Friis equation, augmented with a factor that takes into account the nature of the environment, e.g., indoors versus outdoors, presence of multi-path, etc. Another approach is to rely on calibration: before starting phase 1, the terminals perform measurements in the specific environment where they plan to create a shared secret, in order to assess the behavior of their antennas and receivers within this environment. This calibration needs to happen only once for each specific environment. Since this is the approach we take in our experimental evaluation, we describe it next in more detail.

We set SNR_1 to the minimum SNR value experienced during calibration by any receiving terminal when it is located inside the transmitter's primary lobe. Our rationale is that, whenever a receiving terminal is inside the transmitter's primary lobe, we want the receiver to correctly decode all transmissions. The worst-case scenario related to SNR_1 is that, during phase 1, Bob experiences SNR below SNR_1, even though he is located inside Alice's primary lobe (e.g., because he is located too far from her, or there is an obstacle between them). In this case, Bob will not be able to decode Alice's transmissions, which will reduce the size of the final group secret created in phase 2. This does not affect the security of our system (Eve still knows very little about the created group secret), but, if it happened often, it would reduce the system's efficiency.

SNR_2 is meant to capture the best SNR that Eve may experience when she is located (a) at the minimum physical distance δ from the transmitter, but (b) still outside the transmitter's primary lobe. One approach would be to set SNR_2 to the maximum SNR value experienced during calibration by any receiving terminal that satisfies (a) and (b). In our experimental evaluation, we take a more conservative approach: we put a node (playing the role of Eve) in the best possible location where Eve can position herself while still satisfying (a) and (b), we let the terminals take turns transmitting, and we set SNR_2 to the average SNR experienced by our fictitious Eve. For instance, in our testbed (Figure 1), we place our $n = 4$ terminals at the corners of a rectangle and the fictitious Eve right in the middle, because, to the best of our current understanding, given this environment and this number of terminals, this is the best location where Eve can position herself to overhear the terminals' transmissions (she is on a perfect line with two terminal pairs).

To account for the scenario where Eve has better hardware than our terminals, we increase SNR_2 by multiplying it by a factor of $\frac{\nu_T}{\nu_E}$, where ν_T and ν_E denote the thermal noise power at a terminal's and Eve's receiver. For instance, $\frac{\nu_T}{\nu_E} = 2$ configures the system to be secure in the scenario where Eve's receiver introduces half the thermal noise introduced by any terminal's receiver.

4.2 Estimating Eve's uncertainty

At the end of phase 1, we lower-bound the amount of information that Eve has missed. More specifically, of all the transmitted packets that have been correctly received by at least one terminal, we lower-bound the number missed by Eve. For this, we rely on geometry and the same rationale that we used to estimate SNR_2: We assume that, when a terminal transmits, those inside the primary lobe receive everything while those outside receive nothing. We also assume that Eve has positioned herself in the best possible location to overhear the terminals' transmissions.

We illustrate with an example: Suppose we have $n = 4$ terminals, not positioned on a single line (as illustrated in Figure 1), all of them within each other's transmit range. When a transmitting terminal has positioned its beam such that one other terminal is inside the primary lobe and Eve is outside, we call this a "useful" beam position (because it allows the transmitting and receiving terminals to create secrecy). In our particular example, when a terminal transmits, there are at least two useful beam positions: the transmitting terminal will point its beam once at each of the 3 other terminals, and Eve can be inside the primary lobe in at most one of these beam positions. Hence, of all the packets transmitted by a terminal in phase 1, at least those transmitted during 2 beam positions are received by one other terminal and not Eve. Of course, we do not know which two these beam positions are (because we do not know Eve's location). And this is precisely why we need a secret-agreement protocol like the one described in Section 3, which only needs as input a lower bound on the number of packets missed by Eve, not which particular packets she missed.

The geometric approach that we use requires at least $n = 3$ terminals, not all positioned on a single line and within each other's transmit range. Otherwise, our system conservatively creates no group secret, because it is always possible for Eve to position herself so as to prevent one of the terminals from creating any secrecy with the rest.

5. EVALUATION

In this section, we evaluate our system in terms of the quality of the secrets it generates (how much Eve knows about them) and the rate at which it generates new secrets. We first describe our testbed (Section 5.1) and metrics (Section 5.2), then present our experimental results (Section 5.3).

5.1 Testbed

Our testbed consists of 5 WARP nodes [3], deployed as depicted in Figure 1, on a 7×7 m^2 open terrace. Each node transmits through a flat-panel directional antenna [1] with a 40-degree primary-lobe angle. The transmission power is -19 dBm and the transmission rate 1 Mbps. In each experiment, one node plays the role of Eve and $n = 4$ nodes play the role of the terminals that use our system to continuously create new secrets. In phase 1, all the terminals take turns in transmitting, and in phase 2 these transmissions are used to create a shared group secret. Each experiment corresponds to specific physical locations for the terminals and Eve, and it yields a stream of new secrets agreed on by the terminals.

We selected this particular placement of the nodes, because it allows us to experiment with an adversarial scenario: when the role of Eve is played by the node located in the middle. We call this adversarial for three reasons: (a) Eve is located at 3 m from any terminal, which is closer than any pair of terminals are located to each other. (b) Whenever one terminal points at another, Eve is positioned close to the peak of the main secondary lobe of the transmitter's antenna. This results in the maximum SNR that we measured in this environment for a node positioned outside the transmitter's primary lobe. (c) Eve is positioned on the lines that connect two pairs of terminals (hence prevents them from creating pairwise secrets in phase 1). This results in the maximum fraction of packets overheard by Eve that we measured in this environment. We do not have a proof that this is the worst node placement for our system given this environment, $n = 4$ terminals, and a minimum distance

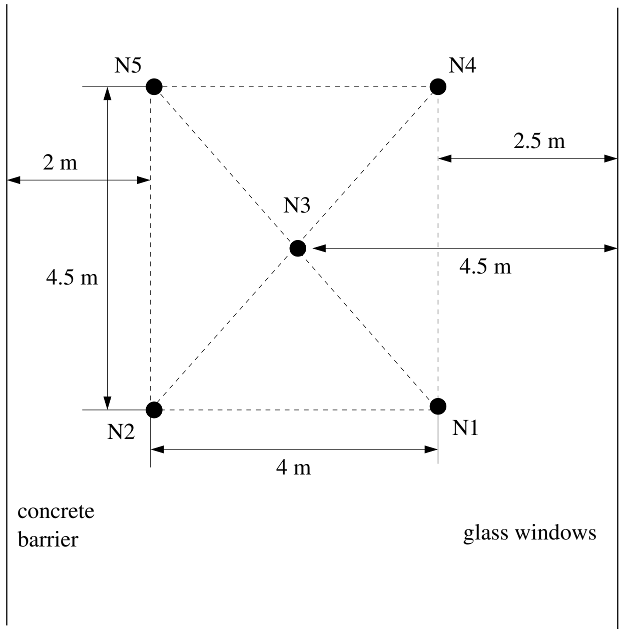

Figure 1: Our testbed: 5 nodes (4 terminals and 1 Eve). The identity of the node that plays the role of Eve changes with each experiment.

of $\delta = 3$ m between any terminal and Eve; but it is the worst that we were able to observe.

This testbed has the following characteristics:

(a) The minimum physical distance between the terminals and Eve is $\delta = 3$ m, which is determined by the shape of the transmitting antenna beam. If Eve can be closer than 3 m to a terminal, she manages to be inside its transmitter primary lobe all the time, and the system is not secure any more.

(b) The maximum physical distance at which the terminals can communicate with each other is about 5 m. If two terminals are further away (e.g., nodes N_1 and N_5), they do not hear each other with a sufficient SNR to decode each other's packets.

(c) We conservatively assume that each transmitting terminal achieves only one useful beam position in phase 1 (Section 4.2). E.g., consider the case where the role of Eve is played by node N_3: Node N_1 exchanges packets with N_2 and N_4 (N_5 is too far and cannot decode N_1's packets). Since Eve's location is not known to the terminals, the system assumes that she may be located either between N_1 and N_2 or between N_1 and N_4. Hence, it conservatively assumes that Eve overheard half of the packets transmitted by N_1 and received by any other terminal.

5.2 Metrics

We consider the following metrics:

- *Reliability* represents how much Eve knows about the secret stream. Reliability q means that Eve can correctly guess each secret bit with probability 2^{-q}. The maximum reliability is 1, which means that Eve can correctly guess each secret bit with probability 0.5, i.e., knows nothing about the secret stream.

- *Efficiency* is the number of secret bits agreed on by the terminals, divided by the total number of bits transmitted by the terminals. By definition, efficiency has to be below 1 (otherwise, every bit transmitted by the terminals would automatically become a secret bit, which is infeasible).

- *Secrecy rate* is the rate at which the terminals agree on new secrets (in bits per second). This does not provide any extra information over efficiency (it is equal to efficiency times the transmission rate of the terminals). We report it only because we find "1 secret Kbps" a more intuitive piece of information than "efficiency 0.001."

We consider two variations of our system:

- *Sequential beamforming*: In phase 1, each terminal transmits in beam positions that differ by 20 degrees from each other (half the angle of the primary lobe). The reason we rotate by half the primary-lobe angle (as opposed to the exact angle) is to ensure that each receiving terminal will be located close to the vertical center of the transmitter's primary lobe in at least one beam position. We use enough beam positions to cover the area of the testbed, and each terminal transmits the same number of packets from each beam position.

- *Selective beamforming*: This is like sequential beamforming, with the difference that, in phase 1, a terminal transmits only when pointing its beam at a receiving terminal, which requires that the terminals know each other's exact position. This variation is designed to achieve better efficiency (avoid useless transmissions) in sparse topologies, where most beam positions are useless, in the sense that they do not point to any receiving terminal.

We compare our system against two non-implementable alternatives, where an oracle helps us configure our system and/or determine Eve's uncertainty:

- *Oracle-Erasures*: This alternative helps us evaluate how well we lower-bound the number of packets missed by Eve in phase 1: it works like our system, with the difference that an oracle tells us the correct number of packets missed by Eve in phase 1. I.e., this alternative achieves perfect reliability and the maximum possible efficiency that can be achieved using our beamforming and wiretap coding with the given SNR_1 and SNR_2 thresholds, and our secret-agreement protocol.

- *Oracle*: This alternative helps us evaluate how well we choose our SNR_1 and SNR_2 thresholds: it works like Oracle-Erasures, with the difference that an oracle tells us the SNR thresholds that—given the traffic exchanged during phase 1—will lead to perfect reliability and the maximum possible efficiency. I.e., this alternative achieves perfect reliability and the maximum possible efficiency that can be achieved using our beamforming and wiretap coding, and our secret-agreement protocol.

5.3 Results

In all experiments, we achieve perfect reliability. This is because we conservatively lower-bound the number of packets missed by Eve in phase 1. However, the more conservative we are in picking this lower bound, the smaller the size of the secret that we create given a certain number of terminal transmissions. So, the question is: how much does our conservatism cost us in terms of efficiency and secrecy rate?

We show the efficiency and secrecy rate achieved by our system and the oracle alternatives, with sequential and selective beamforming, in Figures 2 and 3, respectively. The x-axis corresponds to the identity of the node playing the role of Eve. We make three observations:

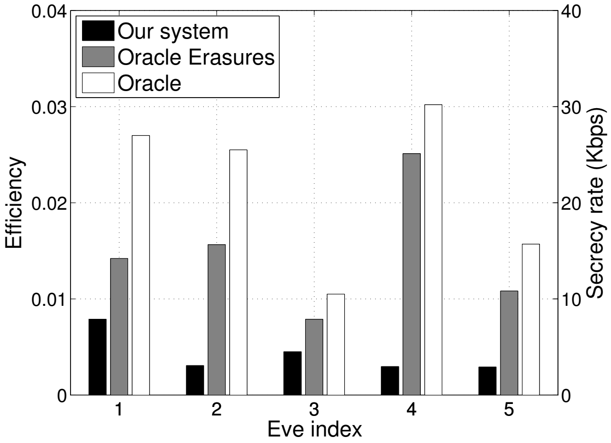

Figure 2: Efficiency with sequential beamforming.

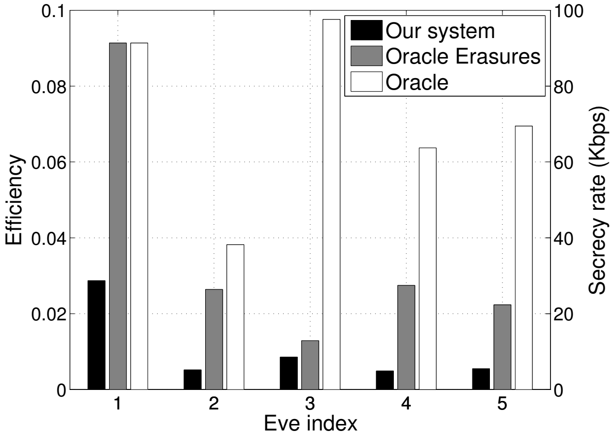

Figure 3: Efficiency with selective beamforming.

First, in all experiments, our system achieves a secrecy rate of several Kbps. As expected, Oracle is more efficient than Oracle-Erasures, which is more efficient than our system. This is because Oracle-Erasures knows exactly which packets Eve has successfully decoded, hence it does not need to conservatively lower-bound this number. Oracle, moreover, knows exactly which packets the terminals and Eve will successfully decode, hence it does not need to conservatively choose the SNR thresholds for the wiretap code. The gap between the efficiency achieved by these alternatives and our system shows the price we pay for being conservative.

Second, when we use sequential beamforming (Figure 2), the efficiency gap between our system and the oracle alternatives is smallest when Eve is the node in the middle. This is not surprising, given that we chose the SNR thresholds for our wiretap code based on this scenario. It illustrates why we cannot achieve high efficiency for all possible positions of Eve: if we want to account for the adversarial scenario (where Eve is in the middle), we necessarily become inefficient in the friendlier scenarios (where Eve is further away from the terminals).

Third, selective beamforming (Figure 2) can significantly improve the efficiency of our system, up to 20 secret Kbps. This is not surprising, given that it reduces useless transmissions. At the same time, selective beamforming increases the efficiency gap between our system and the Oracle alternative. This is due to the fact that Oracle chooses the SNR thresholds for its wiretap code based on a-priori knowledge of who will receive what and when; if we also allow it to leverage this knowledge to altogether avoid useless transmissions (which is what selective beamforming enables), it can pick even better SNR thresholds.

6. RELATED WORK

Wyner introduced the wire-tap channel, where Alice transmits information to Bob, and a “wiretapper” Eve receives a noisy version of what Bob receives; he showed that Alice and Bob can achieve non-zero secrecy rate when the Alice/Bob channel is better than the Alice/Eve channel [30]. Maurer extended the wiretap channel with public discussion; he provided lower and upper bounds for the secrecy rate from Alice to Bob and showed that it can be non-zero even when the Alice/Bob channel is worse than the Alice/Eve channel [21]. This work proved the theoretical feasibility of perfect pairwise secrets (assuming Alice is connected to Eve through an idealized channel). This triggered a significant body of theoretical work [22, 23, 29, 24, 25, 26, 16], which also addressed the scenario of multiple terminals [9].

More recently, researchers have started to propose concrete, implementable protocols for creating pairwise shared secrets in wireless networks. Some of them rely on the time-varying nature of wireless channels and the fact that Alice and Bob can measure the (time-varying) channel between them whereas Eve cannot [13, 17, 7, 32, 8]. Of these protocols, the ones that have been implemented achieve secret generation rates up to a few tens of bps (in modified 802.11 or 802.15 environments). However, they rely on channel changes to extract secret bits, hence they are better-suited for mobile environments (in static environments, they can be vulnerable to eavesdropping) [15].

Our work is more closely related to the following proposals: Alice and Bob can create pairwise shared secrets by combining and heuristically condensing the frames that are transmitted between them only once (the assumption being that these frames are less likely to have been heard by Eve than the rest) [31]. In another proposal, Alice and Bob create pairwise shared secrets by leveraging transmissions made by Alice while pointing a directional antenna at Bob (the assumption being that Eve is not located close to the line between Alice and Bob and, even if she is, she does not always have the computational capability to decode the frames that reach her receiver) [6]. These two protocols have been implemented in an 802.11 environment, but there has not been any evaluation of the reliability or efficiency of these implementations yet. In iJam, when Alice transmits, Bob jams a part of her transmission in a special way (specific to OFDM) that prevents Eve from guessing which part was jammed. Hence, Alice and Bob share common knowledge that is secret from Eve, and they use it to create a pairwise secret [12]. iJam achieves a secret-generation rate up to 18 Kbps (in a modified 802.11 environment).

Our proposal differs from the above protocols in the following ways: First, we create group secrets, whereas the above protocols are fundamentally tied to pairwise secrets. Second, we do not require (a) the existence of quick channel changes in the wireless environment, nor (b) that Eve has limited computational capabilities, nor (c) custom physical-layer operations that are specific to OFDM (or any other transmission scheme).

We build on the idea that a conservative estimate of the number of bits missed by Eve is sufficient for creating shared secrets at Kbps rates, which we first expressed in an earlier workshop paper [28]. We have improved on that work in two ways: First, that work relied on artificial noise to create a secrecy-friendly environment; the problem with artificial noise is that (a) it requires trusted infrastructure to generate it and (b) a powerful adversary may be able to cancel at least some part of it. Instead, we are now using beam-forming, performed by the terminals themselves. Second, our earlier work assumed the existence of erasure channels between the terminals and Eve, i.e., did not account for the fact that Eve may extract useful information from partially received packets. Instead, we are now using wiretap codes to enforce this condition.

7. CONCLUSIONS

We have presented a system that enables n wireless nodes to agree on a group secret $\mathcal{S}$ in the presence of an eavesdropper Eve, such that Eve obtains very little information on $\mathcal{S}$. The novelty of the system lies in the fact that it does not make any assumptions about Eve's computational capabilities, but instead assumes that she has limited network presence. Our system is not yet ready for deployment (we have evaluated it only on one testbed, with one adversary). However, it constitutes the first experimental evidence that it is feasible for a group of wireless nodes to agree on thousands of perfectly secret bits per second, without relying on the adversary's computational limitations.

8. APPENDIX

8.1 Background: Multi-layer Wiretap Codes

A wiretap code, first introduced by Wyner in [30], enables a Bob who has a "better" channel than Eve, to receive securely information from Alice at a rate that depends on how much better his reception SNR is. More specifically, assume that the Alice-Bob and Alice-Eve channels take the fixed values $(h_B, h_E) = (h_1, h_2)$, and thus Bob and Eve experience receive SNRs $SNR_1 = \frac{|h_1|^2}{N_0}$ and $SNR2 = \frac{|h_2|^2}{N_0}$, respectively, where assumed they experience the same thermal power N_0. Then the two channels become simple Gaussian channels of capacity $C_B = \log_2(1 + SNR_1)$ and $C_E = \log_2(1 + SNR_2)$. Assume Bob has a "better" channel than Eve, that is, $SNR_1 > SNR_2$ and $C_B > C_E$. Alice can use a wiretap code to securely transmit a message at any rate $0 < R < C_B - C_E$ so that Bob can decode the message with an arbitrarily small probability of error while Eve is kept totally ignorant of the message. Hence, Bob receives the message essentially "perfectly" while Eve knows nothing about it, it is erased for her. The difference in the quality of the SNR_1 and SNR_2 channels, determines the rate at which we can securely send information.

This is the simplest form of a wiretap code, that is designed specifically for the values SNR_1, SNR_2, and achieves the maximum secure rate for these channels. If we do not know what these values are, we can employ a *layered* wiretap code [19, 18] that leverages the following observation. The wiretap code we designed for the channels SNR_1 and SNR_2 will still yield perfect reception for Bob and an erasure for Eve as long as $SNR_B > SNR_1$ and $SNR_E < SNR_2$. It would not achieve the optimal rate possible, as it is designed to only exploit the SNR_1 and SNR_2 difference, but it would still guarantee erasures. Thus we can think of the SNR_1 and SNR_2 as two thresholds we can impose, that enable us to get secrecy at a fixed rate whenever Bob has a channel realization better than SNR_1 and Eve worse than SNR_2. Such a code is determined by two design parameters: the values of SNR_1 and SNR_2.

To achieve rates that more closely follow the actual differences in the channel realizations of Bob and Eve, we can use instead of two, multiple layers [19, 18]. The more layers we use, we better match the difference in the Bob-Eve realizations, but the more complex the wiretap code becomes. Moreover, the more difficult it also becomes to accurately estimate the distribution of the channel realizations of Eve. Other optimizations are possible to further increase the secrecy rates, such as power allocation optimizations.

8.2 Linear Combinations

Main approach

Our algorithms construct the y-packets, the z-packets and the s-packets as linear combinations of the x-packets. Namely, let X, Y, Z, S, be matrices that have as rows the x, y, z, and s-packets respectively, then

$$Y = A_y X, \qquad Z = A_z Y, \qquad S = A_s Y.$$

During the first phase, Eve observes some of the x-packets, i.e., she observes $W = A_E X$, where the matrix A_E specifies which distinct x-packets are known to Eve. During the second phase we assume that Eve observes all the z-packets, i.e., the matrix Z. In this appendix we provide the constructions for the matrices A_y, A_z and A_s, and prove that they enable information theoretical security as we described in Section 3.

To prove that our constructions are secure, we need to prove that

$$H(S|Z, W) = H(S).$$

We achieve this in steps, by constructing y-packets, z-packets and s-packets that have the following properties:

- The y-packets are constructed to be linearly independent from Eve's observations, and thus information-theoretically secure from Eve. That is, our constructions achieve

$$H(Y|W) = H(Y). \tag{1}$$

- The z-packets that enable reconciliation, are a function of the y-packets. Thus, since the y-packets are linearly independent from Eve's observations, so are the z-packets, and we have

$$H(Z|W) = H(Z). \tag{2}$$

- We construct the s-packets to be linearly independent from the z-packets, since Eve may overhear the z-packets during the reconciliation. Thus, we construct the s-packets to satisfy

$$H(S|Z) = H(S). \tag{3}$$

  Note that since the s and z-packets are a function of the y-packets, we also have

$$H(S, Z|W) = H(S, Z). \tag{4}$$

We then have

$$\begin{aligned} &H(S|Z, W) \\ &= H(S, Z|W) - H(Z|W) \\ &\overset{(a)}{=} H(S, Z) - H(Z) \\ &= H(S|Z) \\ &\overset{(b)}{=} H(S) \end{aligned} \tag{5}$$

where (a) is due to (2) and (4), and (b) is due to (3).

Construction of y-packets

Alice considers each subset of terminals $\mathcal{J}$, identifies the $N^{\mathcal{J}}$ x-packets that were received by all the terminals in the subset but no other terminals, and creates $M^{\mathcal{J}}$ linear combinations $y_1, \dots, y_{M^{\mathcal{J}}}$ of these packets as

$$Y = A_y X,$$

where matrix X has as rows the $N^{\mathcal{J}}$ x-packets, matrix Y has as rows the $M^{\mathcal{J}}$ y-packets and A_y is the generator matrix of a Maximum Distance Separable (MDS) linear code with parameters $[N^{\mathcal{J}}, M^{\mathcal{J}}, N^{\mathcal{J}} - M^{\mathcal{J}} + 1]$ (*e.g.,* a Reed-Solomon code [20]). Each terminal T_i in the set $\mathcal{J}$ can also reconstruct these y packets. We next prove in Lemma 1 that this construction ensures that the y-packets are linearly independent, and Eve has no information about any of them. Alice creates in this manner in total $M = \bigcup_{\mathcal{J}} M^{\mathcal{J}}$ y-packets. Each terminal T_i creates $M_i = \bigcup_{\mathcal{T}_\rangle \in \mathcal{J}} M^{\mathcal{J}}$ y-packets.

LEMMA 1. *Consider a set of N x-packets, say $x_1, \dots, x_N$, and assume Eve has a subset of size $N - M$ of the x-packets. Construct M y-packets, say $y_1, \dots, y_M$, as*

$$Y = AX,$$

*where matrix X has as rows the N x-packets, matrix Y has as rows the M y-packets, and A is the generator matrix of a Maximum Distance Separable (MDS) linear code with parameters $[N, M, N - M + 1]$ (*e.g., *a Reed-Solomon code [20]). Then the M y-packets are information-theoretically secure from Eve, irrespective of which subset (of size $N - M$) of the x-packets Eve has.*

Proof: ▷Let W be a matrix that has as rows the packets Eve has. To prove that the y-packets are information-theoretically secure from Eve, we must show that:

$$H(Y|W) = H(Y).$$

▷ We can write

$$\begin{bmatrix} Y \\ W \end{bmatrix} = \begin{bmatrix} A \\ A_E \end{bmatrix} X \overset{def}{=} BX,$$

where A_E is a $N - M \times N$ matrix of $\text{rank}(A_E) = N - M$, which specifies the $N - M$ distinct x-packets that are known to Eve. A_E is *not known* to us, however we know is that in each row of A_E there is only one 1 and the remaining elements are zero; so all of the vectors in the row span of A_E have Hamming weight (the number of nonzero elements of a vector [20]) less than or equal to $N - M$. On the other hand, from properties of MDS codes, $\text{rank}(A) = N - N_E$, and any set of M columns of the matrix A_y are linearly independent [20]; thus the row span of A and A_E are disjoint (except for the zero vector) and the matrix B is full-rank, i.e. $\text{rank}\,(B) = N$.

▷ If the packets x_i have length Λ, we have that:

$$\begin{aligned} &H(Y|W) = H(Y, W) - H(W) = \\ &= \text{rank}\,(B)\,\Lambda - \text{rank}(A_E)\Lambda = M\Lambda \\ &= \text{rank}(A)\Lambda = H(Y). \end{aligned}$$

■

Construction of z-packets

Alice constructs the z-packets as linear combinations of the Y packets., i.e.,

$$Z = A_z Y,$$

so that: every terminal $T_{i \neq 0}$ can combine $M - M_i$ z-packets with the M_i y-packets it already has and reconstruct all the M y-packets; choosing the z-packets can be done using standard network-coding techniques [11].

Construction of s-packets

Alice constructs the L s-packets using again a linear code, i.e,

$$S = A_s Y,$$

where A_s is constructed using any standard basis-extension method so that

$$\text{rank}\left(\begin{bmatrix} A_s \\ A_z \end{bmatrix}\right) = M.$$

Lemma 2 shows that the s-packets are secure from Eve.

LEMMA 2. *Consider a set of M y-packets, say $y_1, \dots, y_M$, and a set of $M - L$ z-packets, say $z_1, \dots, z_{M-L}$, related as*

$$Z = A_z Y,$$

where matrix Y has as rows the M y-packets, matrix Z has as rows the $M - L$ z-packets, and A_z is a known $M - L \times M$ full rank matrix. Assume that Eve knows all the z-packets. Using any standard basis-extension method [14], find an $L \times M$ matrix A_s, with $\text{rank}\,(A_s) = L$, such that

$$\text{rank}\left(\begin{bmatrix} A_s \\ A_z \end{bmatrix}\right) = M.$$

Then we can construct L s-packets, say $s_1, \dots, s_L$, as

$$S = A_s Y,$$

where matrix S has as rows the s-packets, and satisfy $H(S|Z) = H(S)$.

Proof: To prove that the s-packets are information-theoretically secure from Eve, we need to show that $H(S|Z) = H(S)$. Similarly to the proof of Lemma 1, if the y-packets have length Λ, we have that:

$$\begin{aligned} &H(S|Z) = H(S, Z) - H(Z) = \text{rank}\left(\begin{bmatrix} A_s \\ A_z \end{bmatrix}\right)\Lambda - \\ &- \text{rank}\,(A_z)\,\Lambda = L\Lambda = \text{rank}\,(A_s)\,\Lambda = H(S). \end{aligned}$$

■

8.3 Coded Cooperative Data Exchange

Assume we have n nodes and a set of packets; each node has a subset of the packets, and is interested in collecting the ones she misses. We want to achieve this using the minimum total number of transmissions from the nodes. We can solve this problem in *polynomial time*; for completeness, we provide in the following an Integer Linear Program (ILP).

We will use the following notation:

- $\mathcal{L}$: a subset of the nodes; there exist 2^n such subsets.
- $\mathcal{L}^c$: set of all nodes not in $\mathcal{L}$.
- $\mathcal{P}_i^c$: set of all packets that node i does not have.
- K_i: total number of transmissions that source i makes.

$$\begin{aligned} &\min \quad K_1 + K_2 + \ldots + K_n \\ &\text{subject to} \\ &\sum_{i \in \mathcal{L}} K_i \geq |\bigcap_{i \in \mathcal{L}^c} \mathcal{P}_i^c|, \qquad \forall \mathcal{L}, \\ &K_i \in \mathbf{Z}^+. \end{aligned}$$

9. REFERENCES

[1] DataSheet for the L-com HyperLink HG2418P Antenna. http://www.l-com.com/multimedia/datasheets/DS_HG2418P.PDF.

[2] Redbow Labs. http://vimeo.com/37679492.

[3] Rice University Wireless Open-Access Research Platform (WARP). http://warp.rice.edu.

[4] Shoelace Wireless. http://www.shoelacewireless.com/.

[5] Google Survey: The On-demand Video Consumer. http://www.youtube.com/yt/advertise/research.html, 2012.

[6] Y. Abdallah, L. A. Latif, M. Yousse, A. Sultan, and H. E. Gamal. Keys through ARQ: Theory and Practice. *IEEE Transactions on Information Forensics and Security*, 6(3):737–751, 2011.

[7] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener. Robust Key Generation from Signal Envelopes in Wireless Networks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2007.

[8] J. Croft, N. Patwari, and S. Kasera. Robust Uncorrelated Bit Extraction Methodologies for Wireless Sensors. In *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010.

[9] I. Csiszar and P. Narayan. Secrecy Capacities for Multiterminal Channels. *IEEE Transactions on Information Theory*, 54(6):2437–2452, 2008.

[10] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[11] C. Fragouli and E. Soljanin. Network Coding Fundamentals. *Foundations and Trends in Networking*, 2007.

[12] S. Gollakota and D. Katabi. Physical Layer Wireless Security Made Fast and Channel Independent. In *Proceedings of the IEEE INFOCOM Conference*, 2011.

[13] J. Hershey, A. Hassan, and R. Yarlagadda. Unconventional Cryptographic Keying Variable Management. *IEEE Transactions on Communications*, 43(1):3–6, 1995.

[14] Horn and Johnson. Matrix Analysis. *Cambridge press*, 1985.

[15] S. Jana, S. N. Premnath, M. Clark, S. Kasera, N. Patwari, and S. Krishnamurthy. On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments. In *Proceedings of the ACM MOBICOM Conference*, 2009.

[16] B. Kanukurthi and L. Reyzin. Key Agreement from Close Secrets over Unsecured Channels. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 2009.

[17] H. Koorapaty, A. Hassan, and S. Chennakeshu. Secure Information Transmission for Mobile Radio. *IEEE Communications Letters*, 4:52–55, 2000.

[18] Y. Liang, L. Lai, H. Poor, and S. Shamai. A Broadcast Approach for Fading Wiretap Channels. Submitted to IEEE Transactions on Information Theory.

[19] Y. Liang, L. Lai, H. Poor, and S. Shamai. The Broadcast Approach over Fading Gaussian Wiretap Channels. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2009.

[20] F. J. Macwilliams and N. J. A. Sloane. The Theory of Error Correcting Codes. *North-Holland*, 2006.

[21] U. Maurer. Secret-Key Agreement by Public Discussion from Common Information. *IEEE Transactions on Information Theory*, 39:733–742, 1993.

[22] U. Maurer. Information-Theoretically Secure Secret-Key Agreement by NOT Authenticated Public Discussion. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 1997.

[23] U. Maurer and S. Wolf. Privacy Amplification Secure Against Active Adversaries. In *Proceedings of the International Conference on Cryptology (CRYPTO)*, 1997.

[24] U. Maurer and S. Wolf. Secret Key Agreement over Unauthenticated Public Channels Part III: Privacy Amplification. *IEEE Transactions on Information Theory*, 49(4):839–851, 2003.

[25] R. Renner and S. Wolf. Unconditional Authenticity and Privacy from an Arbitrarily Weak Secret. In *Proceedings of the International Conference on Cryptology (CRYPTO)*, 2003.

[26] R. Renner and S. Wolf. The Exact Price for Unconditionally Secure Asymmetric Cryptography. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 2004.

[27] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[28] I. Safaka, C. Fragouli, K. Argyraki, and S. Diggavi. Creating Shared Secrets Out of Thin Air. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2012.

[29] S. Wolf. Strong Security Against Active Attacks in Information-Theoretic Secret-Key Agreement. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 1998.

[30] A. D. Wyner. The Wire-tap Channel. *Bell System Technical Journal*, 54(8):1355–1387, 1975.

[31] S. Xiao, W. Gong, and D. Towsley. Secure Wireless Communication with Dynamic Secrets. In *Proceedings of the IEEE INFOCOM Conference*, 2010.

[32] C. Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N. Mandayam. Information-Theoretically Secret Key Generation for Fading Wireless Channels. *IEEE Transactions on Information Forensics and Security*, 5:240–254, 2010.