Robust Uncorrelated Bit Extraction Methodologies for Wireless Sensors

Jessica Croft
Electrical and Computer Engineering Department
University of Utah
jessica.croft@utah.edu

Neal Patwari
Electrical and Computer Engineering Department
University of Utah
npatwari@ece.utah.edu

Sneha K. Kasera
School of Computing
University of Utah
kasera@cs.utah.edu

ABSTRACT

This paper presents novel methodologies which allow robust secret key extraction from radio channel measurements which suffer from real-world non-reciprocities and *a priori* unknown fading statistics. These methodologies have low computational complexity, automatically adapt to differences in transmitter and receiver hardware, fading distribution and temporal correlations of the fading signal to produce secret keys with uncorrelated bits. Moreover, the introduced method produces secret key bits at a higher rate than has previously been reported. We validate the method using extensive measurements between TelosB wireless sensors.

Categories and Subject Descriptors

C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: General–Security and protection

General Terms

Security, Experimentation, Measurement, Performance

Keywords

bit extraction, radio channel, RSS

1. INTRODUCTION

For many applications of wireless sensor networks, data privacy is a key requirement. Since sensor nodes may be collecting private data, for example, in patient health monitoring networks, users must have guarantees of privacy. Without data privacy, patients will not be willing to participate and hospitals will not be in compliance with confidentiality regulations. However, because of the limited energy and computational resources of sensor nodes, realistic methods for secure authentication and privacy face special challenges.

To meet the critical need for secure communications, existing research has developed methods to address these multiple challenges. Existing work uses predistributed shared secret keys and public key methods adapted for use on resource constrained sensor nodes. Various methods of probabilistic predistribution [3] [11] have balanced security and limited on-device storage space. Public key methods have used elliptic curve cryptography [12] to create public keys within sensor node resources.

Unlike traditional cryptography methods, we address the problem of secret key establishment between two wireless sensor nodes for secure communication using the time and space variations in the time-division duplex channel. The radio channel offers a unique opportunity to build alternate robust security solutions in a resource efficient manner. A key generated from radio channel characteristics [1] [9] [19] reflects the uniqueness of the time and space in which it was created. Two nodes, Alice and Bob, are able to measure a characteristic of the channel between them, each generates a key from those measurements, and then uses that key to encrypt further communications. Even if Eve, an attacker, were able to overhear legitimate users Alice and Bob during the collection of channel measurements, Eve would be unable to duplicate the key because she would not have measured the same channel as that between Alice and Bob.

Using temporal and spatial variation in channel characteristics for secret key establishment is not a new idea. Key generation from channel characteristics was first described in [7]. Since then several existing efforts including our own have designed and evaluated bit extraction schemes using many different channel characteristics. Some of these characteristics are angle of arrival [1], phase [7] [19], received signal strength [13] [9] [17], signal envelopes [2] [20] and level crossings [13]. Of these, received signal strength (RSS), or channel gain, is most commonly available because of the low device cost and the requirement for inexpensive sensor nodes. To keep the cost low and to be able to use off-the-shelf hardware, we also use RSS in this paper.

Unfortunately, existing methods have significant problems achieving high bit generation rates when required to achieve (1) a low probability of bit disagreement and (2) uncorrelated bits. Existing methods sacrifice bit generation rate to achieve low bit disagreement rates. A low bit generation rate leads to high energy consumption as nodes repeatedly probe the channel to extract sufficient bits. This severely limits the lifetime of the node. The high rate uncorrelated bit extraction (HRUBE) method can achieve a high rate of uncorrelated bits with a reliably low probability of bit disagreement. However, it requires precise knowledge of the distribution and the temporal statistics of the radio channel. Sensor nodes are deployed in a wide variety of environments so such *a priori* knowledge is unrealistic. Further, if statistical assumptions are made that are incorrect, the benefits of the method are lost.

Here we present a method which comprehensively addresses these limitations. Our scheme implements a ranking method to remove the non-reciprocities that are inevitable as a result of wireless sensors having differing transceiver hardware characteristics. Ranking is more robust because even when the measured values at different nodes are of a different scale, the order of the measurements will be the same. For example, the method avoids the disagreements caused by differing transmit powers and RSSI circuit variations. Even in identical hardware, variations of scale exist, and with different hardware, differences will be greater. Ranking also makes the bit extraction process independent of fading distribution. Further, we test and develop protocols which adaptively determine the covariance structure of the measured data in order to reliably extract high entropy rate secret keys with a tunable probability of bit disagreement.

We experimentally test our method using TelosB wireless motes. We evaluate and compare schemes using data collected in three different environments in 25 data sets, totaling 450,000 RSS samples. The extensive data collection allows accurate characterization of important figures of merit, including extracted bits per sample and entropy rate. While the design of a robust and practical scheme is the main objective of this work, we also find that our scheme improves the rate at which secret bits can be extracted. The tested method can extract 40 bits per second at a probability of bit disagreement of 0.04. Compared to the HRUBE bit extraction method, this method is more robust to differences in hardware, adapts to the channel environment, can be implemented on a wireless mote and produces 30% more bits per sample. The tested method produces the highest secret key extraction rate reported to date.

The rest of this paper is organized as follows. Section 2 lays out the adversary model used in this paper. In Section 3 we will describe the Ranking HRUBE method. Section 4 describes our data collection process. In Sections 5 and 6 we address issues related to implementation on wireless sensors. Sections 7 and 8 contain a summary and discussion of our findings. Section 9 forms a conclusion.

2. ADVERSARY MODEL

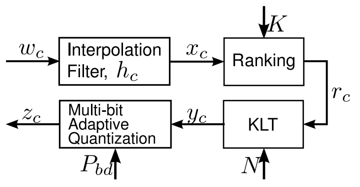
We assume that the adversary, Eve, can listen to all the communication between Alice and Bob. Eve can also measure both the channels between herself and Alice and between herself and Bob at the same time when Alice and Bob measure the channel between them for key extraction. We assume that Eve is more than a few wavelengths away from Alice or Bob. We also assume that Eve knows the key extraction algorithm and the values of the parameters used in the algorithm. We assume that Eve cannot jam the communication channel between Alice and Bob. We also assume that Eve cannot cause a man-in-the-middle attack, i.e., our methodology does not authenticate Alice or Bob. In this aspect, the technique of key extraction from RSS is comparable with classical key establishment techniques such as Diffie-Hellman [4], which also use message exchanges to establish keys and do not authenticate Alice or Bob.

3. METHODOLOGY

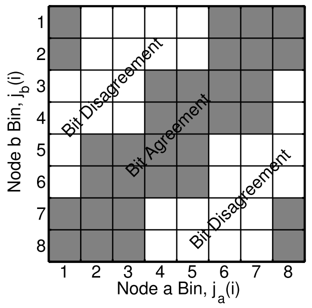
Key extraction benefits from the reciprocity of the channel gain (or loss) between two antennas and the fluctuations of the channel gain in a non-static channel. In a reciprocal channel, the multipath properties including gain, phase shifts and delays are identical in both directions of a link at any point in time. However, successful key extraction must account for the sources of non-reciprocities present in *measurements* of the channel gain, such as additive noise, and differences in hardware. These non-reciprocities are the source of bit disagreement, i.e. bits that do not match between the two generated keys. In addition, a good key has uncorrelated bits, despite the fact that fading is a temporally-correlated random process. The adaptive ranking-based uncorrelated bit extraction (ARUBE) method uses four tools to address these challenges:

1. *Interpolation* removes non-reciprocities caused by the half-duplex nature of the channel.

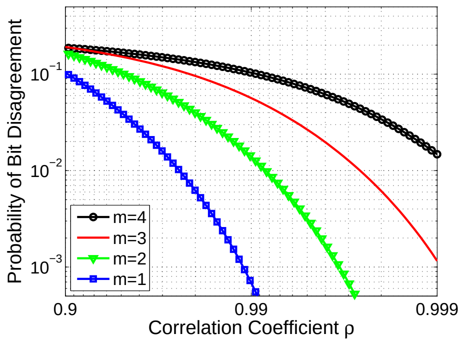
2. *Ranking* reduces non-reciprocities caused by differing hardware characteristics and outputs data with



(a)



(b)



(c)

Figure 1: (a) ARUBE bit extraction. (b) Areas of bit agreement and bit disagreement for $m(i) = 1$. (c) Spatial correlation vs. P_{bd} and m

an *a priori* known distribution.

3. *Decorrelation* removes temporal correlation from the RSS fading signal.

4. *Quantization* extracts bits from interpolated, ranked and decorrelated RSS measurements.

A block diagram is shown in Figure 1(a). We expand upon these steps in the following sections.

3.1 Interpolation

The half-duplex nature of the PHY layer (*e.g.*, in 802.15.4) means that Alice and Bob are unable to simultaneously measure the channel gain. To compensate we use a finite impulse response (FIR) fractional delay filter, which interpolates to obtain an estimate of the channel gains in both directions of the link at a single point in time. The *fractional delay* between the i^{th} measurement by Alice, $w_a(i)$, and the i^{th} measurement made by Bob, $w_b(i)$, is,

$$\mu = \frac{1}{2}\left[\frac{\tau_b(i) - \tau_a(i)}{T}\right] \tag{1}$$

where $\tau_b(i)$ and $\tau_a(i)$ are the arrival times of the i^{th} packet at Bob and Alice respectively.

We implement two fractional delay filters, one each at Alice and Bob. W.l.o.g. we assume that $\tau_a(i) < \tau_b(i)$ so that $\mu > 0$. If we interpolate points in $\mathbf{w}_a$ so that the i^{th} sample is delayed by $(1+\mu)T$ and interpolate points in $\mathbf{w}_b$ so that the i^{th} sample is delayed by $(1-\mu)T$, we would have nearly simultaneous measurements. These delays can be broken down into fractional, μ, and integer, n, delays. At each node:

$$\mu_a = \mu \quad \mu_b = 1-\mu \quad n_a = 1 \quad n_b = 0 \tag{2}$$

We implement the cubic Farrow filter [5]. For $c \in \{a, b\}$:

$$\begin{aligned}\mathbf{h}_c &= \left[\mu_c^3/6 - \mu_c/6, -\mu_c^3/2 + \mu_c^2/2 + \mu_c,\right.\\ &\qquad \left.\mu_c^3/2 - \mu_c^2 + 1, -\mu_c^3/6 + \mu_c^2/2 - \mu_c/3\right]^T\end{aligned}$$

The filtered signal, $\mathbf{x_c}$, becomes the input to the next step in the bit extraction process.

3.2 Ranking

Ranking is used to remove the differences in the unknown transmitter and receiver characteristics which differ between the two directions. As its output ranking also produces values with a uniform distribution.

3.2.1 Motivation

As we note above, the channel gain is reciprocal, but each receiver actually measures RSSI, a voltage in the receiver IC. The RSSI has an affine relationship with channel gain, denoted CG,

$$\text{RSSI} = c_1\text{CG} + c_o \tag{3}$$

and $c_1, c_0 \in \mathbb{R}$ depend on the two nodes. The parameter c_0 will vary due to differing transmit powers or differing battery voltages at the two nodes. Both c_0 and c_1 vary because the devices use different hardware or because manufacturing differences in identical hardware [16].

The device parameters c_0 and c_1 can be considered to be constant over the short periods time required to generate a secret key from the channel (tens of seconds). If the channel gain is reciprocal and the RSSI is given by (3), ranking will recover identical signals.

The ranking process also homogenizes the output distribution. As will be discussed in Section 3.4, it is required to know the distribution of the data input into the quantizer. Ranking does not provide a uniform distribution as input to the quantizer because decorrelation is performed in between ranking and quantization; however, ranking does eliminate the changes that would occur based on the particular environment. For example, narrowband fading statistics may be Ricean, Rayleigh, or Weibull distributed [6], however, the distribution of the output of the ranking operation will remain uniform.

3.2.2 *Algorithm*

Next, we describe how to perform ranking for the ARUBE method. In short, we take each segment of K values from the continuous-valued, interpolated channel measurements and output discrete-valued numbers which indicate their order within the group of K. We also use a set of known "dummy values" to increase the randomness of the output of the ranking. However, for introductory purposes, we first introduce ranking without dummy values, and then define the process of ranking with dummy values.

The input to the ranking operation are the K-length sub-vectors $\mathbf{x}_c^{(t)}$, for $c \in \{a, b\}$. By sub-vectors, we mean that channel interpolated channel measurements, $\{x_c(i)\}_i$, are input to a serial-to-parallel converter that outputs sub-vectors of length K, which we denote $\mathbf{x}_c(t)$. Specifically,

$$\mathbf{x_c^{(t)}} = [x_c((t-1)K+1), \ldots, x_c(tK)]^T \tag{4}$$

Ranking is a function $R : \mathbb{Z}^k \rightarrow \mathcal{K}_0^K$, where $\mathcal{K}_0$ is a set of finite size with minimum 1 and maximum K. When there are no "ties" in input data, $\mathcal{K}_0 = \{1, \ldots, K\}$, and $\mathbf{x}_c(t)$ is ranked such that the j^{th} element of the t^{th} ranked sub-vector is

$$\begin{aligned} r_c^{(t)}(j) &= |\{k : x_c^{(t)}(j) > x_c^{(t)}(k)\}| + 1 \\ &+ \frac{1}{2}|\{k \neq j : x_c^{(t)}(j) = x_c^{(t)}(k)\}| \end{aligned}$$

When there are no ties in the input data, $r_c^{(t)}(j)$ is simply the order of $x_c^{(t)}(j)$ in a sorted list of $\mathbf{x_c^{(t)}}$. When there are ties, the value of $r_c^{(t)}(j)$ is the average of the order of the tied values in the sorted list. For example, for $K = 5$ and this particular x_c, the vector $\mathbf{r}_c$ would be output from the ranking method,

$$\begin{aligned} x_c(i)_i &= [\underbrace{13, 11, 10, 14, 11}_{\mathbf{x}_c^{(1)}}, \underbrace{12, 16, 17, 19, 15}_{\mathbf{x}_c^{(2)}}, 18, 17] \\ r_c(i)_i &= [\underbrace{4, 2.5, 1, 5, 2.5}_{\mathbf{r}_c^{(1)}}, \underbrace{1, 3, 4, 5, 2}_{\mathbf{r}_c^{(2)}}] \end{aligned} \tag{5}$$

If the number of input values of $\{x_c(i)\}_i$ cannot be evenly divided by K, the left over values are not used.

Next we describe the introduction of "dummy values" to add randomness to the output of our ranking method. Ranking the measurements directly introduces non-randomness that could possibly be exploited by an attacker. If the first $K - k$ measurements are known or guessed, for $k \ll K$, it would be less difficult to accurately determine the ranks of the remaining k measurements. To avoid this problem, we introduce D dummy values into the input stream. The ranking with dummy values is a function $R : \mathbb{Z}^k \rightarrow \mathcal{K}_D^K$, where $\mathcal{K}_D$ is a set of finite size with minimum 1 and maximum $K+D$. When there are no ties in input data, $\mathcal{K}_D = \{1, \ldots, K+D\}$.

In the ARUBE method, we determine D dummy values from D evenly spaced quantiles of the distribution of $\{x_c(i)\}_i$. Specifically, we use $F_{x_c}^{-1}\left(\frac{n-0.5}{D}\right)$ for $n = 1, \ldots, D$, where $F_{x_c}(x)$ is the cumulative distribution function (CDF) of x_c. Note that values are found independently at each node $c \in \{a, b\}$.

The j^{th} element of the t^{th} ranked sub-vector, $r_c^{(t)}$, becomes,

$$\begin{aligned} r_c^{(t)}(j) &= |\{k : x_c^{(t)}(j) > d_c^{(t)}(k)\}| + 1 \\ &+ \frac{1}{2}|\{k \neq j : x_c^{(t)}(j) = d_c^{(t)}(k)\}| \end{aligned}$$

where

$$\mathbf{d}_c^{(t)} = \left[\mathbf{x}_c^{(t)T}, F_{x_c}^{-1}\left(\frac{0.5}{D}\right), \ldots, F_{x_c}^{-1}\left(\frac{D-0.5}{D}\right)\right]^T \tag{6}$$

3.3 Decorrelation

Adjacent channel measurements in $\mathbf{r}_c$ are correlated. In this paper we use the discrete Karhunen-Loéve transform (KLT) to convert the measured, interpolated, ranked channel measurements in $\mathbf{r}_a$ and $\mathbf{r}_b$ into uncorrelated components. Given the covariance matrix of correlated data the KLT looks for an orthogonal basis that decorrelates the data. If the data is Gaussian, the decorrelated data will also be independent.

Assume that the input vector at node $c \in \{a, b\}$, $\mathbf{r}_c$, has mean μ_c, covariance matrix R_r and length N. The singular value decomposition (SVD) of R_r can be written, $R_r = USU^T$, where U is the matrix of eigenvectors, and $S = diag\{\sigma_1^2, ..., \sigma_N^2\}$, is a diagonal matrix of the corresponding eigenvalues. We assume that the eigenvectors have been sorted in order of decreasing eigenvalue, so that $\sigma_1^2 \geq \sigma_2^2 \geq ... \geq \sigma_N^2 \geq 0$. Note that $U^T U = I_N$, where I_N is the $N \times N$ identity matrix. The discrete KLT calculates $\mathbf{y}_c$ as

$$\mathbf{y}_c = U^T(\mathbf{r}_c - \mu_c). \tag{7}$$

It can be shown that R_y, the covariance matrix of $\mathbf{y}_c$ is equal to S. Because S is diagonal, $\mathbf{y}_c$ has uncorrelated elements.

In Section 5 we discuss the online determination of R_r and the setting of parameter N.

3.4 Quantization

There is a tradeoff between the probability of bit disagreement, P_{bd}, and the number of bits generated. Multibit adaptive quantization [17] (MAQ) achieves a high rate of bits per sample for a desired P_{bd}.

W.l.o.g. we choose Alice to be the 'leader' and Bob to be the 'follower'. We first quantize $y_a(i)$ into one of $J \triangleq 2^{m_i+2} = 4 \times 2^{m_i}$ equally likely quantization levels. We determine the quantization levels based on the CDF of $y_a(i)$, $F_i(y) = P[y_a(i) \leq y]$. The thresholds, η_j, are calculated as,

$$\eta_j = F_i^{-1}\left(\frac{j}{4 \times 2^{m_i}}\right), \text{for } j = 1, \ldots, J-1. \quad (8)$$

and $\eta_0 = -\infty$ and $\eta_J = \infty$.

The quantization bins are then defined by the thresholds. The j^{th} quantization bin is the interval (η_{j-1}, η_j) for $j = 1, \ldots, J$, so $j(i)$ is given by

$$j(i) = \max_j [j : y_a(i) > \eta_{j-1}] \quad (9)$$

Next, we define the following binary variables:

- Define $e(j)$, for $j = 1, \ldots, J$ as
$$e(j) = \begin{cases} 1, & (j \mod 4) \geq 2 \\ 0, & otherwise \end{cases} \quad (10)$$
- Create a Gray codeword with m_i bits, that is, an ordered list of 2^{m_i} possible m_i-bit codewords.
- Let $f_1(j) = \lfloor \frac{j-1}{4} \rfloor$. Define $d_1(j) \in \{0,1\}^{m_i}$ to be equal to the $f_1(j)th$ Gray codeword.
- Let $f_0(j) = \lfloor \frac{j+1 \mod J}{4} \rfloor$. Define $d_0(j) \in \{0,1\}^{m_i}$ to be equal to the $f_0(j)th$ Gray codeword.

These variables are shown in Table 1 for $m(i) = 1$.

Multi-bit adaptive quantization proceeds as follows. The leader node, Alice in this case, quantizes $y_a(i)$ in the correct quantization $k(i)$ for all components i. Alice then transmits the bit vector $e = [e(j(1)), \ldots e(j(N))]^T$ to the follower node, Bob. Both nodes encode their secret key using codeword d_0 when $e = 0$, and codeword d_1 when $e = 1$. Specifically the secret key for node c is

$$\mathbf{z}_c = [d_{e(j(1))}(j(1)), \ldots, d_{e(j(N))}(j(N))] \quad (11)$$

where $j(i)$ is given in Eq. 9. Figure 1(b) shows a graphic representation of the $m(i) = 1$-bit case.

The P_{bd} in MAQ is related to the correlation coefficient between components and the number of bits extracted from each decorrelated component, $y_a(i)$. The correlation coefficient of the i^{th} component, denoted ρ_i, can be determined from the covariance matrix of the decorrelated components.

$$\rho_i = \sqrt{\frac{[R_y]_{i,i}}{\sigma_i^2}} \quad (12)$$

From the areas of bit disagreement in Figure 1(b), the analytical approximation of bit disagreement rate vs. correlation coefficient in Figure 1(c) is derived [17].

Table 1: $m = 1$ bit MAQ

Bin	Codeword			Interval
j	f_1	f_0	e	of $y(i)$
1	0	0	0	$(-\infty, F_i^{-1}(0.125))$
2	0	0	1	$(F_i^{-1}(0.125), F_i^{-1}(0.25))$
3	0	1	1	$(F_i^{-1}(0.25), F_i^{-1}(0.375))$
4	0	1	0	$(F_i^{-1}(0.375), F_i^{-1}(0.5))$
5	1	1	0	$(F_i^{-1}(0.5), F_i^{-1}(0.625))$
6	1	1	1	$(F_i^{-1}(0.625), F_i^{-1}(0.75))$
7	1	0	1	$(F_i^{-1}(0.75), F_i^{-1}(0.875))$
8	1	0	0	$(F_i^{-1}(0.875), +\infty)$

The greater the correlation between components the more bits that can be extracted or the lower the percentage of bit disagreement. The total number of bits extracted from each group of decorrelated measurements, $\mathbf{y}_c$ is denoted $M = \sum_{i=1}^{N} m(i)$.

4. EXPERIMENTAL DATA COLLECTION

For purposes of evaluation, we implement three wireless sensors capable of collecting RSS measurements. The TelosB mote is a low power wireless sensor module equipped with an IEEE 802.15.4 compliant RF transceiver (the TI CC2420), built-in antenna and a micro-controller.

TinyOS/NesC software is written for the TelosB motes for measurement and communication. Nodes Alice (a) and Bob (b) take turns transmitting probing packets. Each probing packet contains a counter value and a unique node id number. When node $c \in \{a, b\}$ receives the ith packet, it (1) obtains the RSS of the packet, $w_{c,i}$; (2) stores the received counter value i and the RSS value $w_{c,i}$; (3) increments its local counter value and (4) builds a new data packet containing the new counter value and its own node ID and sends it over the radio to node $\bar{c}$ where $\bar{c} \in \{a, b\}$ and $\bar{c} \neq c$.

The packet transmission rate of the device, and thus the RSS sampling rate, is 50 per second. The third node, Eve, designated the attacker node, overhears all of the packets being transmitted between the other two nodes, estimates the RSS of each packet and stores the data. Eve's TelosB mote does not transmit any packets. Data is collected on a laptop to enable arbitrary application of the RSS measurements in secret key establishment.

We collected 25 datasets with a total of 443,600 samples. Most datasets had between 10,000 and 20,000 RSS samples while a few datasets had more than 50,000 or less than 5,000. At 50 samples per second it takes 5 minutes to collect 15,000 samples. The nodes were arranged in various geometries to evaluate the ability of Eve to obtain the same key as Alice and Bob and to see how the signal to noise ratio (SNR) might affect the methods. For all datasets, Alice and Eve were placed on a flat surface while Bob was rotated and moved randomly by an experimenter to introduce random fading into the channel. In the 16 datasets where Eve was present, she was at most 45cm from Alice and in few cases she was less than 6.25cm or $\frac{\lambda}{2}$ from Alice. Six datasets were collected where Bob was more than 1.5m from Alice and Eve. All signal processing was done in Python.

5. ENABLING CHANNEL ADAPTATION

In [17] the authors presented HRUBE, a framework for bit extraction from channel measurements, but did not have a realistic method for implementation. This section presents methods to select the parameters of the ARUBE method. These parameters include the number of decorrelated components, N, the decorrelation matrix, U, and the number of bits per component, $\{m(i)\}_i$. The selection of these parameters depends upon the radio channel between Alice and Bob. For example, in a quickly varying channel we would expect the covariance matrix to be different than in a slowly varying channel. Also, the number of bits extracted from the channel would increase with signal to noise ratio.

5.1 Previous Approach

In the HRUBE method, the covariance matrix, R_x, was estimated as

$$\hat{R}_{x_c,x_c} = \frac{1}{2C-1} \sum_{c\in\{a,b\}} \sum_{i=1}^{C} (\mathbf{x}_c^{(i)} - \hat{\mu}_c)(\mathbf{x}_c^{(i)} - \hat{\mu}_c)^T \tag{13}$$

where $\mathbf{x}_c^{(i)}$ is the i^{th} N-length measured RSS vector at node c, C is the total number of vectors and

$$\hat{\mu}_c = \frac{1}{C}\sum_{i=1}^{C} \mathbf{x}_c^{(i)}. \tag{14}$$

The $N \times N$ decorrelation matrix U is found by the SVD. The values, $m(i)$, were determined from the covariance matrix of $\mathbf{x}_a$ and $\mathbf{x}_b$. The secret key, $\mathbf{z}_c$, was then extracted from the same measurements as were used to estimate the covariance matrix.

5.2 Selection of N

The computational complexity of estimating the covariance matrix and calculating the SVD are both dependent upon N as will be discussed in Section 6. Increasing N will decrease temporal correlation between bits in the secret key because more samples are simultaneously decorrelated. For example, setting $N = 50$ produced sufficiently decorrelated bits for the HRUBE method [17]. Because of the tradeoff between computational complexity and temporal decorrelation, finding a minimum range or value for N could significantly reduce the number of calculations.

In order to test for uncorrelated bits, we look at two types of correlation coefficients:

1. *Pair-wise bit correlation coefficients.* We denote ρ_{z_i,z_j} as the correlation coefficient between the i^{th} and j^{th} component of vector $\mathbf{z}_c$ (Eq 11), for any particular combination (i, j) where $i \neq j$. There are $\binom{M}{2}$ different values of ρ_{z_i,z_j}.

2. *Global bit correlation coefficient.* We denote ρ_z as the correlation coefficient between any pair of different components of $\mathbf{z}_c$. Here we assume that the correlation coefficient is identical across all combinations of (i, j) and we use our data to estimate the single value of ρ_z.

There are $\binom{M}{2}$ different pairwise correlation coefficients, ρ_{z_i,z_j}, but because there are more of them, each one is estimated with few realizations, which we denote as n. The global bit correlation coefficient, ρ_z, is a single number but it has many more realizations, n. By performing statistical tests on both correlation coefficients, we can reliably verify that bits are uncorrelated.

To avoid confusion, it should be noted that we now have two types of correlation, spatial and temporal. The first, spatial, is 'good' correlation (Eq 12 and Figure 1(c)) between the decorrelated components $y_a(i)$ and $y_b(i)$. This spatial correlation is what makes bit extraction effective. The second describes temporal correlation between bits. Both ρ_{z_i,z_j} and ρ_z quantify temporal correlation that might allow an attacker to have a better chance of guessing subsequent bits given knowledge of some bits. We quantify the effect of N on temporal correlation in this section.

Estimated correlation coefficients will never be precisely zero, even if $\rho = 0$. We use hypothesis tests to quantify if these non-zero correlation coefficients are likely to have been generated if the true $\rho = 0$. Formally, the decision is:

$$\begin{aligned} H_0 &: \rho = 0 \\ H_1 &: \rho \neq 0 \end{aligned} \tag{15}$$

The hypothesis test is performed on the t statistic [8],

$$t = \hat{\rho}\sqrt{\frac{1-\hat{\rho}^2}{n-2}} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \tag{16}$$

where $\hat{\rho}$ is the correlation coefficient estimated from the data either ρ_{z_i,z_j} or ρ_z, n is the number of realizations used in the estimate and γ is a threshold. The threshold is set by choosing a desired false alarm rate, α, and applying knowledge of the distribution of t (t distribution with $n - 2$ degrees of freedom). In the limit for high n ($n > 100$) the distribution of t approaches the zero-mean unit-variance Gaussian distribution.

We plot the t-statistics vs. N and the appropriate thresholds for three datasets in Figures 2 and 3. Each dataset has many pairwise correlation coefficients, so for simplicity we plot only the maximum pairwise correlation coefficients in Figure 2. For the datasets presented here, the minimum number of realizations is $n = 833$. We set the false alarm probability, $\alpha = 0.05$, therefore we would expect even if $\rho = 0$ to see 5% of the values crossing the threshold. In all plots the target $P_{bd} = 0.04$, $K = 256$, and $D = 128$.

As shown in Figure 2, for $N \geq 15$ the datasets u, s and t decide H_0 more than $1-\alpha = 95\%$ of the time. The global correlation, ρ_z, as shown in Figure 3, is dependent upon the dataset. H_0 is decided for datasets u, s and t at $N = 27, 25, 17$ respectively. Based on the tests of ρ_{z_i,z_j} we may believe $N > 15$ is sufficient, however, because of the tests on ρ_z, we may wish to set $N > 30$.

We also tested the effect of N on the number of bits extracted per sample. We tested the total number of bits per sample for a range of $5 \leq N \leq 50$ and over the same three datasets. We found that the choice of N does not have a significant effect on the number of bits extracted per sample.

In addition, we tested the entropy of the bitstream vs. N. For N larger than 15, entropy slowly increases with N. These results are presented in Table 6.

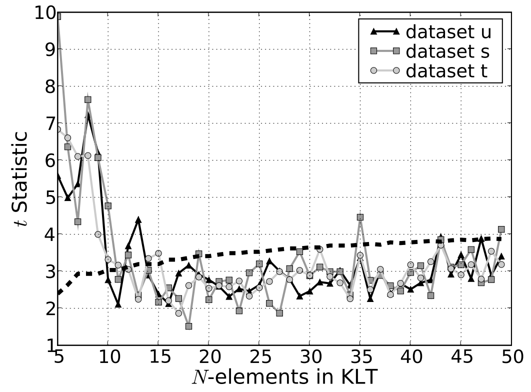


Figure 2: t-statistics for max ρ_{z_i,z_j} vs. N for three datasets and the threshold, γ.

5.3 Covariance Matrix and Correlation Coefficient Estimation

In the previous section we looked at the effect of N on temporal correlation when the covariance matrix was estimated as in Eq. 13. In other words, the covariance matrix was estimated using all measurements made in both directions. If this were implemented, it would take many minutes to collect all of the RSS measurements. Alternatively the covariance matrix would be estimated and the KLT performed for every vector of samples col-

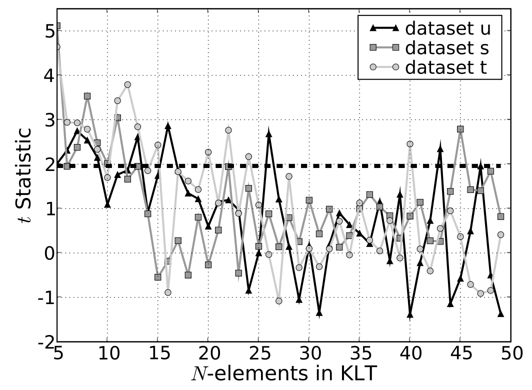


Figure 3: t-statistics for ρ_z, vs. N for three datasets and the threshold, γ.

lected. In either case, it would either computationally expensive or introduce high latency.

We see three options in addition to the *full* method for calculating the covariance matrix:

1. *Full*: The covariance matrix is estimated on the nodes for all vectors of collected channel measurements using Eq. 13. The SVD of the covariance matrix is calculated on each node and the decorrelation matrix, U, is found.

2. *Offline*: The covariance matrix is estimated offline from previously collected data, the SVD of the covariance matrix is calculated and then the decorrelation matrix, U, is loaded onto both nodes prior to deployment.

3. *Uni-directional*: The covariance matrix is estimated by each node using only the measurements it has collected. In this case the covariance matrices at Alice and Bob would be,

$$\hat{R}_{r_a,r_a} = \frac{1}{C-1}\sum_{i=1}^{C}(\mathbf{r}_a^{(i)} - \hat{\mu}_a)(\mathbf{r}_a^{(i)} - \hat{\mu}_a)^T$$

$$\hat{R}_{r_b,r_b} = \frac{1}{C-1}\sum_{i=1}^{C}(\mathbf{r}_b^{(i)} - \hat{\mu}_b)(\mathbf{r}_b^{(i)} - \hat{\mu}_b)^T$$

4. *Partial*: Alice and Bob collect and share N_c preliminary channel measurements, $\mathbf{w}_{pa}$ and $\mathbf{w}_{pb}$. Both vectors are interpolated and ranked then the covariance matrix is estimated at both nodes using the preliminary bi-directional data,

$$\hat{R}_{r_c,r_{\bar{c}}} = \frac{1}{N_c-1}\left[\sum_{i=1}^{N_c}(\mathbf{r}_{pa}^{(i)} - \hat{\mu}_{pa})(\mathbf{r}_{pa}^{(i)} - \hat{\mu}_{pa})^T + \sum_{i=1}^{N_c}(\mathbf{r}_{pb}^{(i)} - \hat{\mu}_{pb})(\mathbf{r}_{pb}^{(i)} - \hat{\mu}_{pb})^T\right] \quad (17)$$

Table 2: t-statistics by method

Method	ρ_{z_i,z_j}		ρ_z	
	N=17	N=3	N=17	N=35
Full	2.950	3.369	1.864	0.444
Offline	2.825	2.194	0.533	1.159
Uni-directional	2.950	3.196	1.978	0.589
Partial $N_c = 1000$	2.201	2.828	0.228	0.926
Partial $N_c = 2000$	2.952	2.851	0.366	1.440

The SVD of the covariance matrix is calculated on each node to obtain U.

The advantages of each method are as follows. The *full* method will decorrelate the measurement vectors better than the other three, but is expensive in terms of time and computation. The *offline* method is much less computationally intensive since the KLT is not calculated online, but does not adapt to changes in the radio channel. The *uni-directional* method requires no additional data sharing between the two nodes other than probe packets and MAQ protocol, but is as computationally expensive as the the full method. The *partial* method, while more computationally expensive than the *offline* method, can adapt to changes in the wireless channel because it decorrelates the bit stream immediately after calculating U.

To determine the effect of these four methods on temporal correlation we take one of the datasets, u, which was also used in the previous section and run the same hypothesis tests. Table 2 shows that none of the four methods results in correlation coefficients ρ_{z_i,z_j} or ρ_z which are significantly different than zero. For all methods, $P_{bd} = 0.04$, $K = 256$ and $D = 128$.

The effect of the covariance estimation method on the bits extracted per sample is also of concern. On average the *partial* method extracted 5% fewer bits per sample than did the *offline*, *full* or *uni-directional* methods. For the *offline* method we used dataset r as the dataset to compute the decorrelation matrix U. Dataset r was collected in similar channel conditions as dataset u.

Rarely, the *uni-directional* method produced as much as 40% fewer bits per sample. This method suffers from the fact that the U matrix can be highly sensitive to noise. This is because the order of the eigenvectors and the sign of the eigenvectors can be different at Alice and Bob. Other methods guarantee U will be identical at both nodes.

To determine the number of bits to extract from each component, Alice and Bob must know the correlation coefficients $\rho(i)$ (Eq. 12). In the *uni-directional* method, Alice and Bob cannot determine the correlation coefficients. In addition, in the *offline* method the values of the correlation coefficients are virtually certain to vary with differing channel conditions. In these two cases, Alice and Bob could do one of two things:

1. Make a conservative guess based on a metric like signal to noise ratio.

2. Exchange a subset of the decorrelated components, $\mathbf{y}_c$, and use them to calculate the correlation coefficients similar to the *partial* method.

Although it would be cheaper both in terms of computation and time if the SVD was calculated offline, it would leave the nodes without any means of calculating a new U matrix or correlation coefficients if the nodes were deployed in an environment with significantly different wireless characteristics than the previously gathered samples. To allow adaptation, we use the *partial* method in the rest of this paper.

6. ARUBE PROTOCOL

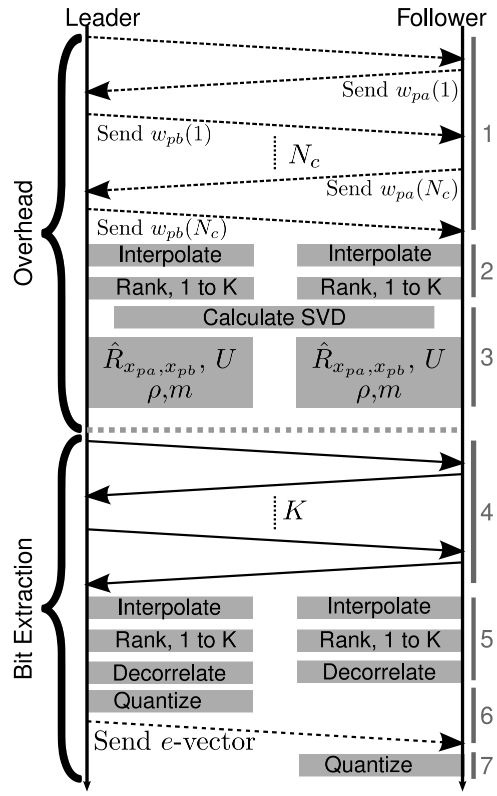


Figure 4: Packets sent for channel probing (—>) and data transfer (- - ->), computation (boxes) at either node, for overhead and bit extraction.

In this section we describe the ARUBE protocol and find the number of transmissions necessary to extract a secret key of length L_k. Figure 4 shows a diagram of the protocol.

At a high level, the protocol has two parts separated by the dotted horizontal line in Figure 4. In the first part (steps 1-3 in Figure 4) the two nodes estimate the covariance matrix and calculate the decorrelation matrix, U, and the bit vector, $\mathbf{m}$. In the second part (steps 4-7) the nodes measure the channel and using U and m, extract bits for a secret key. The second part can be repeated as many times as necessary to obtain the desired number of bits in the secret key. The process can be described as follows:

1. Alice (the leader) and Bob (the follower) exchange N_c packets. The packets contain the RSS value of the last received packet at the respective node so that both nodes have a copy of the preliminary RSS measurement vectors.
2. Alice and Bob rank and interpolate both vectors.
3. Both nodes estimate bi-directional covariance matrix, calculate the SVD to find the decorrelation matrix, U, and the bit vector, $\mathbf{m}$.
4. Alice and Bob exchange K probing packets which contain no data. After packets are exchanged, Alice has a vector of RSS as measured from Bob to Alice and Bob has a vector of RSS as measured from Alice to Bob.
5. Alice and Bob interpolate, rank and decorrelate their RSS vectors to obtain $\mathbf{y}_a$ and $\mathbf{y}_b$ respectively.
6. Alice quantizes $\mathbf{y}_a$ to obtain the secret key, $\mathbf{z}_a$, and the e-vector. She sends the e-vector to Bob.
7. Bob, upon receipt of the e-vector from Alice, quantizes $\mathbf{y}_b$ to obtain the secret key $\mathbf{z}_b$.

The fourth through seventh steps are performed until the secret key is of desired length. If the channel changes substantially or the percentage of bit disagreement is higher than expected, the first three steps can be performed again to obtain an estimate of current channel statistics.

With the ARUBE protocol in mind we determine the number of transmissions needed to create a shared secret key of length L_k. We define the constants

$$\begin{aligned} N_c &= \text{Samples required to calculate } \hat{R}_{r_{pa},r_{pb}} \\ N &= \text{Length of vector to be decorrelated} \\ K &= \text{Number of samples to rank} \\ B_e &= \text{Bits extracted per sample} \end{aligned}$$

We calculate the number of transmissions required to generate a key of length L_k and the computational complexity of each step with respect to N, K and N_c. The number of bits extracted per sample, B_e, is dependent upon the environment where the bit extraction is performed.

Table 3: Number of Packets Transmitted

B_e	Node	Overhead	Key 1	Key 4	Key 7
0.4	Alice	1000	1263	2052	2841
	Bob	1000	1256	2024	2792
0.75	Alice	1000	1264	1800	2336
	Bob	1000	1256	1768	2280

6.1 Packet Transmissions

Table 3 shows the number of packets transmitted when $L_k = 128$, $N_c = 1000$, $K = 256$ and $B_e = [0.4, 0.75]$ as the number of keys created increases. The number of packets transmitted is

$$N_t = N_c + \left(\left\lceil \frac{L_k}{B_e K} \right\rceil K + G \right) \tag{18}$$

Where G is the number of packets required for Alice to transmit the e-vector. G is dependent on the number of bits in a packet, P, and the number of components in $\mathbf{y}_c$ from which bits can be extracted $M_n = |\{i : m(i) \neq 0\}|$.

$$G = \frac{L_k}{M} M_n \frac{1}{P} \tag{19}$$

The number of bits extracted per sample, B_e, has the greatest effect on the number of packets transmitted. The transmissions above the dotted horizontal line in Figure 4 are overhead and are independent of the number or length of secret keys to be generated. The amount of transmission overhead is dependent only upon N_c. While the leader and follower nodes transmit nearly the same number of packets, the leader node will transmit more over time because of the e-vector packets.

6.2 Computational Complexity

The gray boxes in Figure 4 indicate computations that are done on each respective node. The computational complexity of each step is listed in Table 4.

While the calculation of the SVD has the highest order of any operation, it may be possible to simplify the order. For example only $M_n = |\{i : m(i) \neq 0\}|$ of eigenvectors need to be calculated. If $M_n \leq N$ it can be less computationally complex to calculate one eigenvector at a time and stop extracting eigenvectors when $m(i) = 0$. Depending upon the number and length of keys to be generated, the covariance matrix estimation and calculation of the SVD might not be the most significant portion of the required computation although they have the highest order.

Although an exact comparison is difficult, we expect ARUBE to extract secret bits with fewer computations in comparison to the Diffie-Hellman secret key exchange. The main computation for the Diffie-Hellman scheme is the modular exponentiation, $(g^a \mod p)^b \mod p$ [14]. Here, p is a large prime number, g is the generator of

Table 4: Computational Complexity

Overhead	Complexity
Interpolate	$\mathcal{O}(N_c)$
Rank	$\mathcal{O}(N_c log K)$
Calculate $\hat{R}_{x_{pa},x_{pb}}$	$\mathcal{O}(N^2 N_c)$
Calculate SVD	$\mathcal{O}(N^3)$
Bit Extraction	**Complexity**
Interpolate	$\mathcal{O}(K)$
Rank	$\mathcal{O}(K log K)$
Decorrelate	$\mathcal{O}(NK)$
Quantize	$\mathcal{O}(K)$

the order of $p-1$, in the group $< \mathbb{Z}_p^*, \times >$, and a and b are the secrets of Alice and Bob, respectively. This modular exponentiation has a time complexity of $\mathcal{O}(nM(k))$ where n is the number of bits in p, k is the number of bits in a or b, and $M(k)$ is the complexity of a chosen multiplication algorithm. Using the Karatsuba algorithm for multiplication [10], $M(k) = O(k^{1.585})$. The time complexity of the ARUBE bit extraction steps is $\mathcal{O}(NK)$. Considering k and K to be constant, and noting that a smaller symmetric key is equivalent in strength to a much larger Diffie-Hellman Key (e.g., 112-bit symmetric key is equivalent to 2048-bit Diffie-Hellman key [15]), ARUBE is computationally more efficient than the Diffie-Hellman key exchange.

7. RESULTS

In this section we quantify the performance of the ARUBE method. We look at three metrics: (1) secret bits per sample; (2) estimated entropy rate of secret key bits; and (3) resistance to a passive attack.

Secret Bits per Sample: The number of secret key bits generated per sample directly impacts the latency and energy efficiency of key establishment. Figure 5 plots ARUBE (and for comparison, HRUBE) secret bits per sample vs. P_{bd} for $N \in \{17, 35\}$, $K \in \{128, 256\}$, and $D = \frac{K}{2}$. We assume the *best case* the HRUBE method, that it estimates the U and $\{m(i)\}_i$ on the same data set which it then uses to extract bits. Out of 25 data sets, we plot the average of the top three with respect to bits extracted per sample, the average of the bottom three and the average remaining 19 datasets.

We show a comparable analysis with the same datasets for a bit extraction method developed by Mathur et al. [13] in Table 5. Unlike ARUBE, this method was developed solely to produce keys with $P_{bd} = 0$, with no expectation of information reconciliation. This method finds extrusions in a filtered vector of RSS measurements. An extrusion is where the values of a filtered RSS vector are above some threshold γ or below $-\gamma$. If an extrusion is at least m measurements long and exists on both directions of the link, it will be assigned as a 1 if it is above γ, or as a 0 if it is below $-\gamma$.

Table 5: Bits per sample–Mathur et al.

$P_{bd} \leq$	0.0	0.0025	0.01	0.04	0.07
Best	0.074	0.077	0.082	0.088	0.089
Middle	0.055	0.064	0.072	0.074	0.076
Worst	0.0	0.032	0.05	0.057	0.057

Table 6: Average and Minimum Entropy Rates.

	$N = 17$		$N = 35$	
Method	Mean	Min	Mean	Min
ARUBE	0.9808	0.9653	0.9833	0.9757
HRUBE	0.9767	0.9433	0.9825	0.9712

To find the values in Table 5 we selected many values of γ between $0.1\sigma \leq \gamma \leq 1.5\sigma$ where σ is the standard deviation of the filtered RSS vector, and found the maximum bits per sample that could be generated which had a P_{bd} less than a given value. Table 5 shows the average for the best three, worst three and remaining 19 datasets. While this method requires much less computation than ARUBE and unlike similar extraction methods produces keys with high entropy, the number of bits extracted per sample is very low. Even at small P_{bd}, ARUBE produces 4 times more bits per sample and up to 9 times more with larger P_{bd}.

Entropy Rate: We estimate the entropy rate of the generated secret key bits, *i.e.*, a quantification of the uncertainty of the bit sequence. If generated bits are perfectly independent, they should achieve an entropy rate of 1. Although it is not sufficient for a secret key to have a high entropy in order to be secure, it is necessary. We generate bits from datasets using $P_{bd} = 0.04$, $K = 256$, and $D = 128$, and then estimate the entropy rate using the approximate entropy test in the NIST's statistical test suite for random number generators [18]. The average and minimum values over 23 of the 25 datasets are listed in Table 6. The remaining two datasets had < 500 bits, not enough to estimate entropy.

Evaluation of Possible Attacker Success: In this paper we take a straight-forward, if simplistic, view of the ability of an eavesdropper to obtain Alice and Bob's secret key. We provide one way to see how the ARUBE and HRUBE methods perform when under attack from a passive listener. For both methods, Eve performs bit extraction in the same manner as Alice and Bob. Eve overhears the N_c preliminary measurements and the RSS values contained within the packets sent between Alice and Bob to find U and $\{m(i)\}_i$. We assume Eve knows the constants N, K and P_{bd} that Alice and Bob use for bit extraction. The average percentages of bits Eve gets correct for the HRUBE and ARUBE methods over the 16 datasets (where Eve was present)

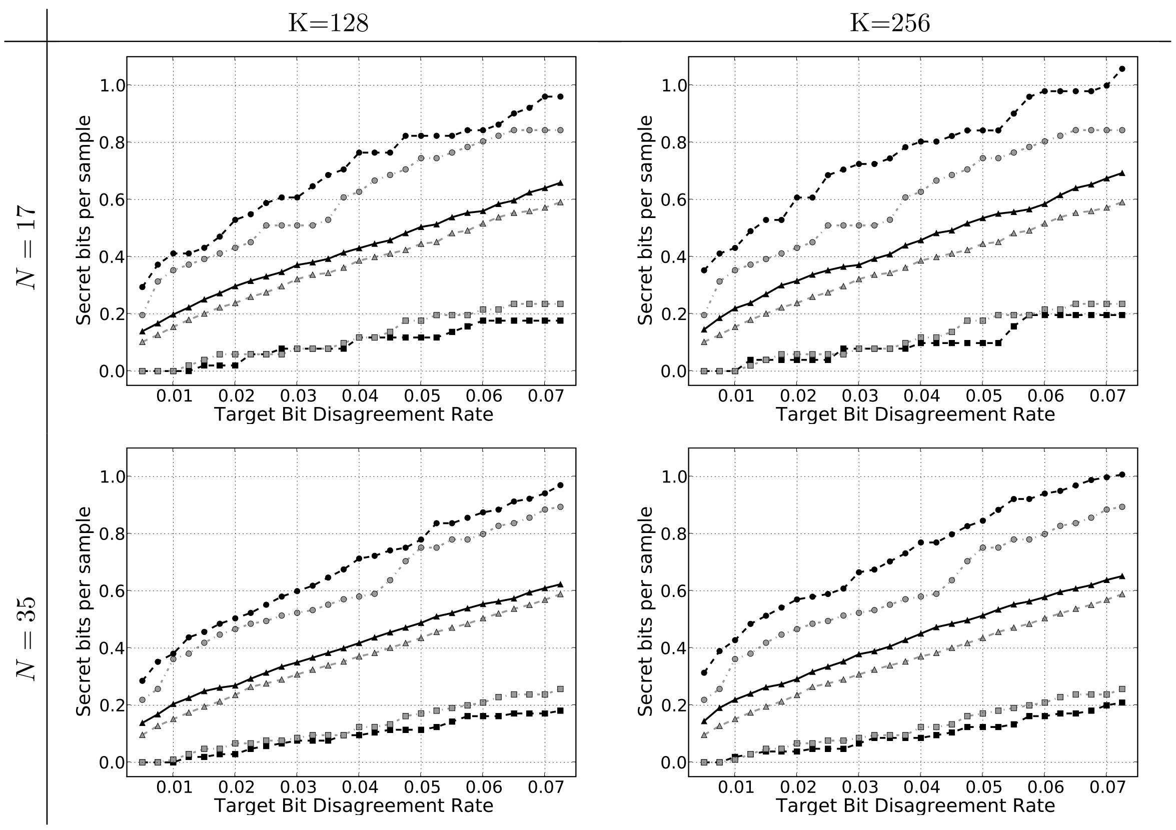


Figure 5: Target P_{bd} vs. secret key bits per sample for ARUBE (black lines) and HRUBE (gray lines), for $N \in \{17, 35\}$, $K \in \{128, 256\}$, and $D = \frac{K}{2}$, for averages of the best three datasets (-●-), the worst three (-■-), and the remaining 19 (-▲-).

Table 7: Percentage of bits Eve gets correct.

Method	Compared to Alice	Compared to Bob
ARUBE	50.19	50.53
HRUBE	50.64	50.76

are compared in Table 7.

8. DISCUSSION

Assuming the best case for the HRUBE method, that it estimates the U and $\{m(i)\}_i$ on the same data set which it then uses to extract bits, we see that the ARUBE still outperforms the HRUBE. Both the ARUBE and HRUBE methods are resistant to a passive evesdropper, as shown in Table 7. The ARUBE method achieves higher entropy than the HRUBE method, and increasing N from $N = 17$ to $N = 35$ also increases the estimated entropy rate for both methods (Table 6).

ARUBE generates up to 60% more bits compared to HRUBE method (Figure 5) for low P_{bd}. For $K = 256$ and $D = 128$, the ARUBE achieves up to 25% more bits for medium and high P_{bd}. For most datasets, the ARUBE achieves higher bit rate at a given P_{bd}. The greatest improvements occur in datasets with high SNR. The performance improvement is seen for both $N = 17$ and $N = 35$. We note that setting K too low reduces the benefit of the ARUBE method, *e.g.*, for $K = 64$ the two methods are approximately equivalent.

Note that K can be set to an arbitrary integer. For instance, if $B_e = 0.8$ and the desired key length is 128 bits, it would be faster to collect and rank $K = \frac{1}{0.8} * 128 = 160$ samples. After U is determined, at 50 samples per second, it would take a wireless sensor 3.2 seconds to collect the required 160 samples for the secret key.

9. CONCLUSION

We presented a new method of secret key generation, ARUBE, that adapts to the radio channel environment and the characteristics of the two wireless sensors in use. Further, for medium and high SNR channels, the ARUBE produces more bits per sample, thus reducing the number of transmissions (energy) required to produce a given length secret key. In comparison with the HRUBE, another uncorrelated bit extraction method, ARUBE extracts 30%-60% more bits in situations with high SNR. ARUBE is shown to produce uncorrelated bits, is resistant to a simple passive eavesdropper, and secret keys have an entropy rate above 0.97. The number of packet transmissions and computational complexity are presented.

Future work should test simplifications and implementations of ARUBE. Algorithms to reduce the computational complexity of the KLT exist and should be tested. The *offline* version of ARUBE is implemented in TinyOS, and current work is implementing the complete method.

10. REFERENCES

[1] T. Aono, K. Higuchi, T. Ohira, B. Komiyama, and H. Sasaoka. Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. *Antennas and Propagation, IEEE Transactions on*, 53(11):3776–3784, 2005.

[2] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener. Robust key generation from signal envelopes in wireless networks. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 401–410, Nov. 2007.

[3] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *In IEEE Symposium on Security and Privacy*, 2003.

[4] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.

[5] C. Farrow. A continuously variable digital delay element. In *IEEE International Symposium on Circuits and Systems, 1988.*, pages 2641–2645, 1988.

[6] H. Hashemi. The indoor radio propagation channel. *Proceedings of the IEEE*, 81(7):943–968, 1993.

[7] J. Hershey, A. Hassan, and R. Yarlagadda. Unconventional cryptographic keying variable management. *IEEE Trans. Commun.*, 43(1):3–6, Jan. 1995.

[8] W. W. Hines, D. C. Montgomery, D. M. Goldsman, and C. M. Borror. *Probability and Statistics in Engineering 4th ed.* John Wiley & Sons, 2003.

[9] S. Jana, S. 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 15th annual international conference on Mobile computing and networking*, pages 321–332. ACM, 2009.

[10] A. Karatsuba. The complexity of computations. In *Proceedings of the Steklov Institute of Mathematics*, volume 211, pages 169–183, 1995.

[11] D. Liu, P. Ning, and W. Du. Group-based key predistribution for wireless sensor networks. *ACM Transactions on Sensor Networks*, 2008.

[12] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Sensor and Ad Hoc Communications and Networks, 2004*, pages 71–80, 2004.

[13] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 128–139. ACM New York, NY, USA, 2008.

[14] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC, 1996.

[15] National Institute of Standards and Technology. *Special Publication 800-57: Recommendation for Key Mangement.* 2007.

[16] N. Patwari and P. Agrawal. *Localization Algorithms and Strategies for Wireless Sensor Networks*, chapter Calibration and Measurement of Signal Strength of Sensor Localization. IGI Global, 2009.

[17] N. Patwari, J. Croft, S. Jana, and S. Kasera. High rate uncorrelated bit extraction for shared secret key generation from channel measurements. *IEEE Transactions on Mobile Computing*, Jan 2010. http://span.ece.utah.edu/uploads/hrubeJournalr1.pdf.

[18] A. Rukhin et al. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. *NIST Special Publication*, pages 800–822, 2001.

[19] A. Sayeed and A. Perrig. Secure wireless communications: Secret keys through multipath. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3013–3016, 2008.

[20] M. Tope and J. McEachen. Unconditionally secure communications over fading channels. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 1, 2001.