Extracting Secret Key from Wireless Link Dynamics in Vehicular Environments

Xiaojun Zhu*†, Fengyuan Xu†, Edmund Novak†, Chiu C. Tan‡, Qun Li† and Guihai Chen*§
*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210024, China
†Department of Computer Science, the College of William and Mary, Williamsburg, VA 23185, USA
‡Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
§Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai Jiao Tong University, Shanghai 200240, China
Email: gxjzhu@gmail.com, {fxu,liqun}@cs.wm.edu, ejnovak@email.wm.edu, cctan@temple.edu, gchen@nju.edu.cn

***Abstract*—A crucial component of vehicular network security is to establish a secure wireless channel between any two vehicles. In this paper, we propose a scheme to allow two cars to extract a secret key from RSSI (Received Signal Strength Indicator) values in such a way that nearby cars cannot obtain the same secret. Our solution can be executed in noisy, outdoor vehicular environments. We also propose an online parameter learning mechanism to adapt to different channel conditions. We conduct extensive real-world experiments to validate our solution.**

I. Introduction

Vehicular networks have gained considerable research interest recently given their potential in improving public safety and traffic management [1]–[3]. Security is an important component in any vehicular network, and one of the most fundamental security requirements is the ability to establish a secure channel between two arbitrary cars. Rather than relying on a public key infrastructure (PKI) based solution [4], in this paper, we propose an alternative approach to allow two arbitrary cars to establish a secret key for secure communications. Our approach can be used in environments where a PKI has not been established. In fact, our approach does not even require the deployment of any special road-side wireless infrastructure.

This is done by having both cars continuously sampling the wireless link and then extracting a shared secret key based on the signal strength fluctuations. The security of the key relies on the unpredictable nature of wireless dynamics, which cannot be observed by an adversary at a distance more than half the wavelength of the wireless signal (e.g., that is 6.25cm for 2.4GHz wireless channel). To sample channel dynamics, two parties capture RSSI values by sending packets back and forth to each other. This will require addressing the following challenges that have rendered the previous approaches [5]–[8] unusable.

First, vehicular environments have very short *channel coherence time*, the time duration for which the wireless channel remains unchanged, due to rapid environment change. Measurements [9], [10] have shown that channel coherence time in vehicular environments can be as short as a few hundred microseconds. Although short coherence time will give high randomness for key extraction in low speed mobile environments [5]–[8], very short coherence time in vehicular environments will pose a big challenge to key extraction. This is because, to sample the same channel dynamics, the two pairing packets sent by the two parties must be within a duration of coherence time. Due to the half-duplex nature of current wireless platforms, however, we find that the round-trip time of a wireless packet may be longer than the coherence time. Applying existing solutions to vehicular environments results in unsatisfactorily slow key generation.

Second, we choose RSSI because it is widely available among off-the-shelf 802.11 radios so that our solution can be readily implemented on existing wireless platforms without hardware changes. But RSSI has a poor accuracy in characterizing channel condition, compared to the Channel Impulse Response (CIR) measurements [5]. We refer to both the short-coherence-time effect and the RSSI error as noise due to the inability to distinguish them.

The fundamental issue in this project is to reduce the effect of noise in our vehicular traces. After filtering out the slow variation [5], which are caused by distance changes, we observe that the noise is very strong, comparable to the level of fluctuation due to the channel dynamics. For high quality traces considered by previous research, the noise level is negligible compared to the signal fluctuation, which is not the case in our trace. Thus, we have a dilemma. If we keep a large portion of the fluctuation, i.e., do not filter out noise sufficiently, the mismatches of bits at the two sides will become so numerous that the number of final extracted bits will be zero after subtracting the effort to correct such mismatches. On the other hand, if we remove too much of the fluctuation, even though mismatches are reduced, the final bits will contain little randomness. Therefore, the main obstacle of this paper is to delineate the fine line between noise and fluctuation of the channel dynamics.

To the best of our knowledge, we are the first to consider key extraction in vehicular environment using RSSI values. None of the previous methods [5]–[8] can be used directly in extracting secret bits in vehicular environments. We have achieved a bit rate around 5 b/s, which is much higher than the previous 1 b/s with the same wireless card platform [5]. Our contributions are as follows. (1) We propose a systematic way to ensure the randomness of the resulting key. With this approach, we know the rate at which we can extract entropy bits. We also select a randomness extractor to actually extract perfect random bits. The extracted bits pass NIST test [11]. (2) We propose an online parameter learning scheme to adaptively adjust parameters which offers more steady performance in dif-

The work was done when the first author was visiting the College of William and Mary.

ferent environments. (3) We propose weighted sliding window smoothing to reduce noise. The novelty of this technique is that it can smooth out local noise, and also compensate for the mismatched sensing time due to the half-duplex nature of existing wireless platform. Experiments show that this technique can improve the performance 50% more than non-smoothing case. (4) Our experiments are conducted using data collected from real world environments.

II. Related Work

Several researchers have proposed to use the unpredictability of the radio channels to extract secret keys. Azimi-Sadjadi et al. [12] first propose an extraction scheme based on signal envelopes and evaluate it against a Rayleigh fading channel model. Later both Mathur et al. [5] and ABSG [7] leverage the channel's level crossings to generate secret keys shared between two parties in the indoor and low-speed outdoor environments respectively. Group key extraction is considered in [13], and the key extraction in [14] is for securing implantable medical devices. Other key extraction approaches [8], [15] are also proposed specifically for the mote platform. There has also been much work on protecting WLANs and sensor networks [16]–[19]. However, neither of them considered key extraction in vehicular networks, where the radio channel has unique characteristics.

Eliminating the dependency among measured samples is critical for the security of the key extraction, but unfortunately the methods applied in prior work do not handle it well. The sub-sampling method in [5], [6] discards many samples along with useful mutual information. That information is precious to our key extraction scheme due to short window of channel sampling time. HRUBE [8] tries to utilize all samples through de-correlation process. The main drawback is that the results, though uncorrelated, may still be dependent. In contrast, we analyze the sample dependency and split the sample sequence accordingly into independent sub-sequences. Therefore we do not waste captured channel information nearly as much.

Additionally, we propose using a deterministic randomness extractor for entropy condensing, rather than the random hashing scheme applied in the previous work [7]. This is because the efficiency (the number of output bits to the number of input bits) of random hashing extractors is upper bounded by the min-entropy of the input bits, and it is inappropriate to use NIST test for validation on the final key because the seeds of random hashing are from an additional random source.

III. Background

Our focus is to extract a shared secret key for two moving cars. We assume that the two cars have a wireless channel to exchange messages when they meet, and they have the ability to record the RSSI reading of the exchanged messages. The two cars, Alice and Bob, exchange messages over wireless channel and record RSSI readings of each message. Any third party, Eve, cannot infer any information about the key. We consider the passive adversary model where the adversary has complete knowledge about the procedure and the exchanged messages. This model is the same as [7], [8].

The general key establishment process is as follows. Here, we assume Alice initiates the process. Alice keeps sending indexed probes to Bob, and Bob immediately returns back acknowledgements (ACKs) upon reception. The probes and ACKs are made as short as possible to deal with the short-coherence-time issue. Both sides record the RSSI reading of received packets along with their indices. Denote by $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ the RSSI readings observed by Alice and Bob respectively. Alice and Bob will extract random bits from X and Y respectively. We also refer to X and Y as *raw data* or *raw readings*. A classic method for key extraction is level crossing [5]. Next, we will introduce the principals behind level crossing and its limitations, followed by our techniques to address these limitations.

A. Level crossing: overview and limitations

Level crossing consists of two steps. First, Alice and Bob keep probing the channel and collecting RSSI readings. They map each reading to a temporary bit as follows. Consider the case for Alice. Let μ_X be the mean of X, σ_X be the standard deviation. Each reading x is mapped to a temporary bit via a quantizer Q such that Q(x)=1 if $x > q_+$; Q(x) =0 if $x < q_-$; Q(x) =e, otherwise, where e is a undefined state and $q_+ = \mu_X + \alpha\sigma_X$, $q_- = \mu_X - \alpha\sigma_X$ where α is a parameter to be tuned. This can be seen in Figure 2. Bob's quantizer is similar.

Second, Alice and Bob communicate to get final bits from temporary bits. They identify *excursions* in temporary bits. An excursion is a consecutive of 1s or 0s with length at least m where m is the second parameter to be tuned. For example, if the temporary bits are "$e0000111e111e$" and $m = 3$, then there are 3 excursions: $e\overbrace{0000}\overbrace{111}e\overbrace{111}e$. Alice finds from her temporary bits all excursions, and sends the indexes of all excursion centers to Bob. On receiving the indexes, Bob checks each index to see whether he also has an excursion around this index, and sends the result back to Alice. Finally, they quantize each common excursion to a bit. For the above example, if Bob has the same temporary bits, then they both get 011. We name the final bits (a bit vector) as *quantized bits*. The bits at Alice and Bob may be different. Each different bit is a *mismatch*. The ratio of the number of mismatches and the number of quantized bits is defined as *mismatch rate*.

The level crossing algorithm subtracts a windowed moving average from the raw data to reduce the influence of slow variation, where the output of the sliding window is the slow variation. This creates the third parameter: the window size s. The residuals, rather than the raw readings, are fed into the quantizer Q. In summary, we have three parameters: reading threshold α, excursion threshold m, and window size s.

To determine the effectiveness in a vehicular environment, we implemented the two most relevant methods, level crossing [5] and ABSG [7]. Due to the noise in a vehicular environment, the ABSG method applied to our trace leads to a very high mismatch ratio, which renders the secret extraction rate not satisfactory. Our experiments find that the generic level crossing method [5] can result in low mismatch ratio. Therefore, we mainly improve on the generic level crossing method. However, directly applying level crossing is not sufficient to achieve good bit rate. Even though level crossing is a groundbreaking approach to secret extraction, it has some questions that are not answered. Applying it to our vehicular trace shows the following problems.

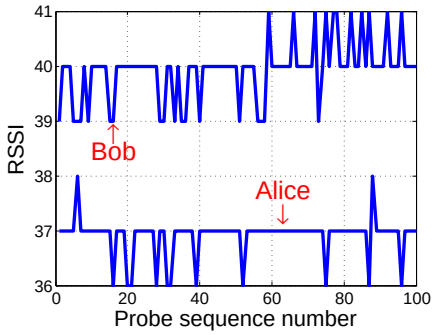


Fig. 1. High noise in vehicular environment. RSSI of Alice changes in a different way as Bob. When Alice goes 1db down, Bob usually remains the same, or even goes 1db up.

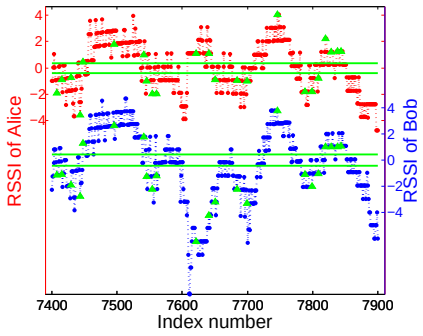


Fig. 2. Level crossing on a portion of residuals. The two lines for Alice (top) or Bob (bottom) are the q_+, q_- in quantizer Q, and each triangle corresponds to a quantized bit.

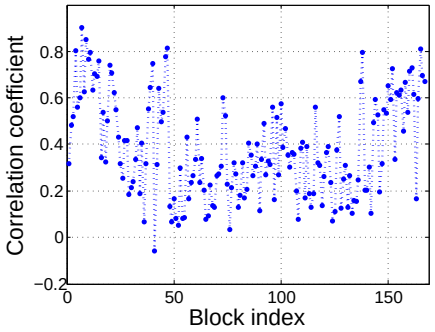


Fig. 3. Non-stable channel condition. Each block corresponds to 2-second residuals. Correlation coefficient changes over time and it reflects the noise level.

First, we observe high noise level in vehicular environments. To illustrate, we plot 100 raw readings from a collected vehicular trace in Figure 1. (The process of trace collection is elaborated in the experiment section.) We can see that the noise level is high. Indeed, the residuals (after subtracting slow variations from raw readings) for this trace have a low correlation coefficient of 0.52. We have to design schemes to remove the noise; otherwise it causes mismatches which seriously influence the performance.

Second, there are no guidelines to select parameters for level crossing. In our experiments we notice that the performance is very sensitive to parameter setting. Setting parameters, however, is hard due to the noise level comparable to channel dynamics. We need to find the fine line to separate the noise and the channel fluctuation so that the mismatch ratio is low and the generated bits will contain high randomness.

Third, the level crossing scheme does not estimate the randomness of the generated bit string nor does it generate a bit string that is necessarily random. We show one such case in Figure 2. In the figure, the quantized bits are not random (readings are consecutively below the lower threshold or over the upper threshold). They are dependent. The original paper [5] employs subsampling on quantized bits to counteract dependency. We find, however, random subsampling can discard many important bits containing high randomness, resulting in slow key generation. Instead, we aim to find an optimal and controllable method to remove dependency among the generated bits.

Fourth, the level crossing scheme uses a fixed set of parameters, which does not adapt to the drastic change of channel dynamics in vehicular environments. Figure 3 shows the correlation coefficient of residual readings at different time periods (2-second units). We observe that the correlation varies at different times, which suggests that a fixed set of parameters may not work well.

B. Overview of our approach

We organize our solution into the framework shown in Figure 4. After sampling, we smooth the raw readings to reduce noise. Our technique is weighted sliding window smoothing. The novelty is that Alice and Bob cooperate to choose the weights such that the resulting smoothed readings

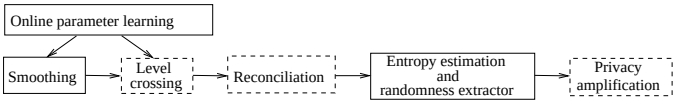


Fig. 4. Flowchart of the system. We focus on the three steps within solid boxes, each of which is a section in this paper.

have maximum correlation. The smoothed readings are fed into the level crossing algorithm, which produces quantized bits.

However, the quantized bits at Alice and Bob may be different. These different bits, i.e., mismatches, are corrected by information reconciliation. We implement Cascade [20], a classical information reconciliation method. During information reconciliation, there will be exchanged parity bits that are exposed to the adversary. This exposed information will be removed by privacy amplification.

After information reconciliation, Alice and Bob have the same quantized bits (otherwise, they will restart the process). But the bits may not be secure due to dependency. To deal with this issue, Alice (Bob) extracts random bits from her (his) own quantized bits. This is done by using a deterministic randomness extractor based on a Markov model of the quantized bits. Besides extracting random bits, they also estimate the entropy contained in the quantized bits. To evaluate the randomness of their extracted bits, we use the NIST test [11], the state-of-the-art statistical test tool for randomness.

Now the two sides have perfect random bits, but the bits are still not secure because the information reconciliation step leaks some information. We apply privacy amplification [21] to distill such information. The final key is secure.

To cope with the non-stable channel condition, we propose an online parameter learning scheme. In the three parameters of level crossing, we choose to learn the reading threshold α online. The scheme can tolerate non-stable channel condition.

C. Performance metric

To determine how well our scheme works, we define our metric *approximate entropy bit rate* as

$$a_{bps} = E \cdot (n_q - n_p)/t$$

where E is the estimated Shannon entropy per bit of quantized bits, n_q is the number of quantized bits, n_p is the number of exchanged parity bits during reconciliation, and t is the time duration of the trace. Note that this metric has taken into account the leaked information during reconciliation.

We also define several related terms. First, *quantized bit rate*: $q_{bps} = n_q/t$. It is the raw rate of level crossing, and does not take into account the leaked information. Second, since we select a randomness extractor to actually extract the entropy, we have *secret bit rate*: $s_{bps} = (n_e - n_p E)/t$ where n_e is the number of extracted bits by our randomness extractor. It does take into account the leaked information. We have $s_{bps} \leq a_{bps} \leq q_{bps}$. The metric a_{bps} reflects the entropy bits per second. Our current implementation can achieve s_{bps}, and in theory it can be improved to approach a_{bps}. For all three bit rates, there are associated mismatch rates. The mismatch rate for q_{bps} has been defined previously. The mismatch rates for a_{bps} and s_{bps} are defined as the percentage of remaining mismatches in the quantized bits after information reconciliation. For mismatch rate, we use a single notation γ_{mis} for three different rates. It can be judged from context which bit rate it is corresponding to.

IV. Smoothing

Vehicular networks operate in very noisy environments which will severely affect the performance of any key establishment algorithm. We propose to use weighted sliding window smoothing to address the issue of noise.

For a fixed window size k (to be determined in experiments), we assign different weights $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_k)$ to readings at Alice and Bob respectively. Let $\sum a_i = 1$ and $\sum b_i = 1$. The ith smoothed reading is $x'_i = \sum_{j=1}^{k} a_j x_{i+j-1}$ (Alice) $y'_i = \sum_{j=1}^{k} b_j y_{i+j-1}$ (Bob). Note that if $\mathbf{a} = (1/k, 1/k, \ldots, 1/k)$ and $\mathbf{b} = (1/k, 1/k, \ldots, 1/k)$, then this is the standard sliding window smoothing scheme. Ideally, we want to find an $\mathbf{a}$ and a $\mathbf{b}$ to maximize our metric, a_{bps}. This problem is hard to solve because the a_{bps} is computed after a series of operations, each of which is not describable by a simple function. We turn to another objective, to maximize the correlation coefficient of resulting bit sequences. Let $X' = (x'_1, x'_2, \ldots, x'_{n-k+1})$ and $Y' = (y'_1, y'_2, \ldots, y'_{n-k+1})$. Then our objective is

$$\max_{\mathbf{a},\mathbf{b}} \quad \rho_{X',Y'} \tag{1}$$

We solve the problem by transforming it into *canonical correlation analysis (CCA)* [22]. Given two matrix $A \in \mathbb{R}^{n\times d_1}$ and $B \in \mathbb{R}^{n\times d_2}$, CCA focuses on finding a linear combination of A's column vectors and a linear combination of B's column vectors such that the correlation coefficient of the two new vectors is maximized. The linear combination for A involves d_1 coefficients (there are d_1 column vectors), and the linear combination for B involves d_2 coefficients.

Our transformation is as follows. Let $\mathbf{x}_i^j$ denote the column vector $(x_i, x_{i+1}, \ldots, x_j)$ and $\mathbf{y}_i^j$ defined similarly. Let $A = \left[\mathbf{x}_1^{n-k+1}, \mathbf{x}_2^{n-k+2}, \ldots, \mathbf{x}_k^n\right]$ and $B = \left[\mathbf{y}_1^{n-k+1}, \mathbf{y}_2^{n-k+2}, \ldots, \mathbf{y}_k^n\right]$ where $A, B \in \mathbb{R}^{(n-k+1)\times k}$. Applying CCA on A and B yields two linear combinations. The linear combination for A gives the optimal $\mathbf{a}$, and the linear combination for B gives the optimal $\mathbf{b}$. Now we have the optimal solution to problem (1). The computation involves eigenvalue decomposition to a matrix with size $k \times k$. Given that k is usually very small in our case, it is computational efficient to solve the problem.

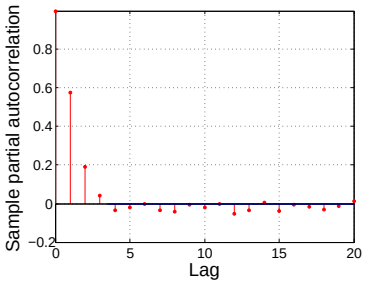


Fig. 5. Partial correlation coefficient.

V. Entropy estimation and randomness extractor

After using level crossing to generate quantized bits, we apply information reconciliation to correcting the mismatches. Then each side has the same quantized bits, and will solve the dependency issue in the quantized bits. In the following, we take the quantized bits from one trace as an example.

A. Dependency modeling

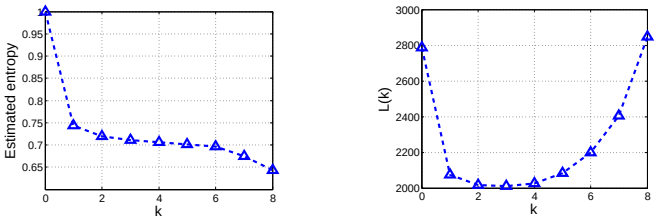
We first show that quantized bits have limited dependency (each bit depends on finitely many previous bits), then we use a Markov chain to model the dependency.

To show the limited dependency property, we plot in Figure 5 the partial autocorrelation coefficient (pacf) [23] of the quantized bits of one trace. Intuitively, $pacf$ describes the correlation between two bits after eliminating the influence of bits in-between. (Note that it is different from autocorrelation where the influence is not subtracted.) For $lag \geq 4$, the absolute value of $pacf$ is very small, meaning each bit only depends on the previous 3 bits. Therefore, quantized bits have finite dependency, we can model it using a Markov chain.

Generally, suppose each bit only depends on the previous bit, then there is one bit memory. The memory can be recorded as $S^{(1)} = \{0, 1\}$. We call this the state space of Markov chain. If each bit depends on two previous bits, then we can use $2nd$ order Markov chain to model it. Its memory space is $S^{(2)} = \{00, 01, 10, 11\}$. Generally, if each bit depends on the previous k bits, then the state space is $S^{(k)}$. We should estimate the order since it is unknown in practice.

This order is estimated by the popular BIC (Bayesian Information Criterion) Markov order estimator [24]. It works as follows. Let n be the total number of bits. Let $\mathbf{s}_1^j = (s_1, s_2, \ldots, s_j)$ and $N(\mathbf{s}_1^j)$ be the number of occurrences of substring $\mathbf{s}_1^j$ in the bit string. Then the estimated order k is the number that minimizes the objective function $L(k)$

$$\min_{k} \quad L(k) = -\log P_{ML(k)} + 2^{k-1} \log n \tag{2}$$



(a) Estimated entropy (b) L(k)

Fig. 6. Estimated entropy and likelihood and for quantized bits of one dataset.

where $P_{ML(k)}$ is the kth order maximum likelihood $\log P_{ML(k)} = \sum_{\mathbf{s}_1^{k+1} \in S^{(k+1)}} N\left(\mathbf{s}_1^{k+1}\right) \log \hat{p}(\mathbf{s}_1^{k+1} \mid \mathbf{s}_1^k)$ where $\hat{p}(\mathbf{s}_1^{k+1} \mid \mathbf{s}_1^k)$ is the empirical conditional probability of string $\mathbf{s}_1^{k+1}$ given $\mathbf{s}_1^k$. We find from experiments that the order of our quantized bits is always between 1 and 10. Thus we can solve (2) by enumerating $L(k)$ at all points and pick the optimal k. In situation where the order is beyond this range, we can also use iterated grid search as in Section VI.

B. Entropy estimation

The entropy of quantized bits is the upper bound of the number of extracted bits [25] and is used in calculating leaked information during reconciliation. We need to estimate the entropy. By definition, for kth order Markov chain, its entropy rate is $H = -\sum_{i \in S^{(k)}} \pi(i) \sum_{j \in S^{(k)}} p(i,j) \log p(i,j)$ where $\pi(\cdot)$ is the stationary distribution and $p(i,j)$ is the transition probability from state i to j. We will show that the estimated order (by BIC) is appropriate for computing entropy.

In theory, for estimating the entropy, we may assume any finitely large order that is higher than the actual order. This is because, any kth order Markov chain is a special case of the mth order Markov chain for $m \geq k$. Consequently, for bits generated by a kth order Markov chain, any mth order ($m \geq k$) Markov chain gives the correct entropy estimation.

However, this result holds only in scenarios where there are infinitely many samples, which may not be satisfied in practice. One solution is to repeatedly increase the order by one and find where the estimated entropy converges. Following this idea, for the same quantized bits as in Figure 5, we show in Figure 6(a) its entropy computed by assuming different orders. The entropy is roughly the same around $k = 2, 3, 4$. The concave part ($k > 4$) is caused by sample size limitation. On the other hand, the objective L (of BIC estimator) in Figure 6(b) gives the same conclusion. The conclusion also coincides with Figure 5. Thus, we are confident about the order estimated by BIC estimator.

C. Randomness extractor

We consider extracting perfect random bits from the dependent quantized bits. It is not safe to simply perform random hashing based on our estimated entropy, which is Shannon entropy, because random hashing should be based on *min-entropy* [26]. For extracting perfect random bits from Markov chain, there is a simple and elegant method [27]. However, its efficiency is low and cannot be improved to approach Shannon entropy. Thus we turn to an earlier approach [25], which is more complex but the efficiency can approach Shannon entropy. The following shows how to extract perfect random bits from a weak random source.

The randomness extractor consists of two steps. First, the bits are split into subsequences as follows. Suppose the Markov chain generating the bits has order k. Then the number of resulting subsequences is 2^k, with each subsequence corresponding to a distinct Markov state. A bit belongs to the subsequence corresponding to the bit's previous Markov state, the state determined by the bit's previous k bits. This splitting process can be done by scanning the bits from left to right. Figure 7 gives an example of splitting a sequence generated by a 2nd order Markov chain into $2^2 = 4$ subsequences, S_{00}, S_{01}, S_{10} and S_{11}. A dash box starts from the leftmost two bits and slides toward the right side one bit per step. In each step, the two bits inside the dash box shows one of the four subsequences, indicating the right-hand bit outside the dash box belong to that subsequence. This procedure repeats until the dash box reach the end of the sequence. It can be proved that each resulting subsequence of the splitting procedure contains independent bits, thus eliminating dependency.

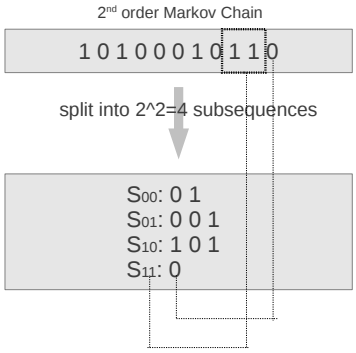


Fig. 7. Independent subsequence splitting. For instance, bits 11 in the dash box indicates that the current state is 11 so that the next bit, 0, belongs to the subsequence S_{11}.

Second, we extract *unbiased* bits from each subsequence. A bit is *biased* if the probability of it being 1 is not equal to the probability of it being 0. Biased bits create patterns that can be exploited by the adversary. Without loss of generality, we assume that the bits are biased. Given a fixed length N of bits, let the number of 1s be k. Since these bits are independent, all the $\binom{N}{k}$ possible cases happen with equal probability. Thus, we can encode each of these $\binom{N}{k}$ cases with $\log \binom{N}{k}$ bits on average. N is a direct factor influencing the efficiency. This encoding process is implemented by an encoding table that maps every N biased bits to a variable length code. The table can be constructed off-line. Though there are many tables for a given N, it does not matter which table is used as long as Alice and Bob use the same table.

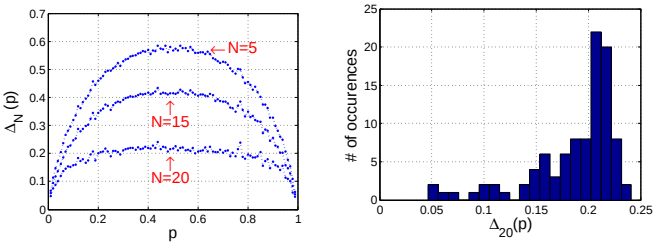
Algorithm 1 illustrates the randomness extractor. We first estimate the order of the Markov model, then split the input into independent subsequences. The last step is to extract unbiased bits from each subsequence and append them together(Lines 5-9). Note that the output length is a random variable, since the code lengths are different for some inputs.

The efficiency of this randomness extractor is defined as the expected number of extracted unbiased bits over the total

Algorithm 1: Randomness extractor

Input: q_bits, the quantized bits after information reconciliation at Alice (Bob); $code\_table$ the encoding table; N, the block size
Output: $bits$, random bits

begin
  let $\hat{k}$ be the optimal solution to objective (2);
  $seqs \longleftarrow$ split q_bits into $2^{\hat{k}}$ different subsequences;
  $bits \longleftarrow \emptyset$;
  foreach $seq \in seqs$ **do**
    $blocks \longleftarrow$divide seq into blocks with each containing N bits;
    foreach $block \in blocks$ **do**
      $code \longleftarrow code\_table(block)$;
      append $code$ to $bits$;



(a) Efficiency gap (b) Histogram for $\Delta_{20}(p)$

Fig. 8. Efficiency of the extractor. For each p=0.01,0.02,. . .,0.99, we simulate 4000 independent random bits. We extract random bits with Algorithm 1. When N approaches infinity, the efficiency gap approaches zero.

number of biased bits. It describes how much randomness is extracted. Its upper bound is the Shannon entropy denoted by $H(\cdot)$. Let p be the probability of a bit being 1 and $\eta_N(p)$ be the efficiency. We have $\lim_{N\to\infty} \eta_N(p) = H(p)$ where $H(p)$ is the Shannon entropy. In practice, we are also interested in the gap between finite N and infinity. Denote the gap by $\Delta_N(p) = H(p) - \eta_N(p)$. We simulate the gap as shown in Figure 8(a). It decreases with the increase of N. Our implementation uses $N = 20$, whose histogram is in Figure 8(b). For simplicity, we refer to $\Delta_{20}(p)$ as Δ.

D. Discussion

One concern of our Markov model is how well it approximates the ground truth. However, the ground truth is unknown. To prove its validity, we have observed the finite dependency property and provide other empirical evidence.

First, each subsequence produced by the first step of our randomness extractor should contain independent bits, thus the bits should be uncorrelated. This is confirmed by Figure 9, which plots the autocorrelation of one subsequence from our experiments. Second, the empirical efficiency of our randomness extractor should be consistent for both simulated data and real trace. We compute Δ for one dataset under different parameter settings (α, m, s) according to $\Delta = \frac{a_{bps} - s_{bps}}{q_{bps}}$. (This can be derived by noting that $n_e = (E-\Delta)n_q$.) Figure 10 plots the histogram, which coincides with the result from simulation

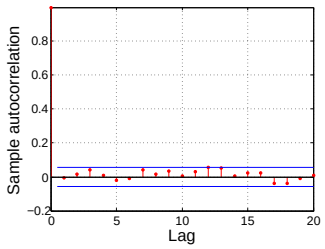


Fig. 9. Autocorrelation of a subsequence. Bits are uncorrelated within 95% confidence interval.

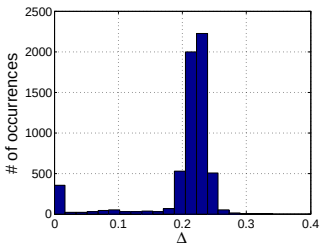


Fig. 10. Empirical efficiency gap. Each occurrence is for a different α, m, s.

in Figure 8(b). Third and most importantly, the resulting bits of the randomness extractor should be able to pass the NIST test. As shown in our experiments, all of our extracted bits pass the test. (Without our extractor, none of them can pass the test.)

VI. Online Parameter Learning

Recall that three parameters should be determined in level crossing. Our experiments show that the excursion threshold m can be set to 2 and the sliding window size s can be the number of samples in 2 seconds. In this section we aim to find a suitable quantization threshold α in an online fashion.

In the online scheme, suppose Alice is the leader. For a training period, Bob sends his residual readings to Alice. Alice simulates the key generation process on her own readings and Bob's readings with different α. Then she picks, and sends to Bob, the α that optimizes the resulting bit rate. After training period, both sides will use this α for key generation. But how to optimize α and which training period should be used?

A. Choosing α

Our ultimate goal is to maximize a_{bps}. One natural solution is to try several α, and compare the resulting a_{bps}. Due to the intermediate steps, such as entropy estimation and randomness extraction, this method is computationally intensive. Instead, we give the following metric f_τ as a qualitative approximation of a_{bps}. For a given α, the leader (Alice or Bob) simulates level crossing with α, obtaining the corresponding quantized bit rate q_{bps} and its mismatch rate γ_{mis}. The metric f_τ maps each (q_{bps}, γ_{mis}) pair directly to a scalar:

$$f_\tau(q_{bps}, \gamma_{mis}) = \begin{cases} q_{bps} \cdot (1 - \gamma_{mis}/\tau) & \text{if } \gamma_{mis} < \tau \\ 0 & \text{otherwise} \end{cases}$$

where τ is a scaling parameter. This metric can be computed quickly for a certain α without performing entropy estimation and information reconciliation. We design f from empirical study. The intuition is as follows. On one hand, if the mismatch rate of quantized bits is above τ, then all the bits will be exposed during information reconciliation, resulting in zero a_{bps}. On the other hand, if the mismatch rate is zero, then we can use all the quantized bits. For any mismatch rate in-between, linear interpolation is performed. The parameter τ is closely related to information reconciliation. We find that $\tau = 0.20$ performs well in our settings. Figure 11 shows the two metrics $f_{0.2}$ and a_{bps} with respect to different α. The optimal α based on f_τ is very close to that based on a_{bps}.

Algorithm 2: Iterated grid search

Input: $data$, residual readings for Alice and Bob; $(low, high)$ low and high bounds on α; n_t, the number of iterations

Output: α, the optimal value that maximizes f

begin
  $[x_0, x_2, x_4] \longleftarrow [low, (low + high)/2, high]$;
  $[y_0, y_2, y_4] \longleftarrow [f(x_0), f(x_2), f(x_4)]$;
  $ite \longleftarrow 1$;
  while $ite \leq n_t$ **do**
    $[x_1, x_3] \longleftarrow [(x_0 + x_2)/2, (x_2 + x_4)/2]$;
    $[y_1, y_3] \longleftarrow [f(x_1), f(x_3)]$;
    Find i such that $y_i = \max\{y_0, y_1, \ldots, y_4\}$;
    $[x_0, x_2, x_4] \longleftarrow [x_{i-1}, x_i, x_{i+1}]$;
    $[y_0, y_2, y_4] \longleftarrow [y_{i-1}, y_i, y_{i+1}]$;
    $ite \longleftarrow ite + 1$;
  $\alpha \longleftarrow x_2$;

Observe that f is approximately unimodal, which can be used to expedite search. To avoid exhaustive search, we employ an iterated grid search algorithm that is based on a three-point scheme [28], as described in Algorithm 2. It compares three points in each iteration and chooses the best one, and then reduces the search range by $2/5$.

The number of f_τ evaluations is $3 + 2 \cdot n_t$ where n_t is the number of iterations. The optimal α resides within a range of $0.6^{n_t} L$ where L is the length of the search range, thus the error bound is $\frac{0.6^{n_t}}{2} L$. In experiments, we set $\alpha \in [0, 2]$ and $n_t = 6$, giving error less than 0.05. The algorithm incurs 15 function evaluations, which is significantly less than exhaustive search.

B. Training period

Choosing different training periods may give quite different performance. We consider two schemes in this subsection. The first is a naive scheme called single-frame scheme. Time is divided into frames, each of which has length $\overline{f}$, in units of the number of readings. In each frame, the leader uses the first $\overline{f} \cdot \gamma_r$ ($\gamma_r < 1$) readings for training, and obtains the best α. Then applies it to the remaining readings of the frame. This scheme does not work well— we cannot find proper γ_r and $\overline{f}$ to give reasonable a_{bps}, which is presented in the experiment section. The reason is that the channel condition is not stable. Thus, the optimal α selected by the training period may not work well in the remaining period.

We then propose a double-frame scheme. We have an α for each frame and choose the bigger of the two. Because the metric f_τ is not symmetric with respect to the optimal α, underestimation and overestimation have different impacts. Specifically, *overestimation hurts less than underestimation*. For a given distance to the peak, underestimation reduces a_{bps} more significantly. To achieve overestimation, we can raise α but also we can lower the threshold τ, even though it may not reflect a_{bps} well. Figure 11 shows the f with $\tau = 0.12$. It does not fit as well as $\tau = 0.20$, but its peak is still close.

To incorporate the CCA component, we add the weights estimation part before training α. After estimation, Alice will send the weights together with the α to Bob. They will use the bigger α and the corresponding weights in the remaining period for generating bits. We name this procedure "double-frame scheme with CCA incorporated".

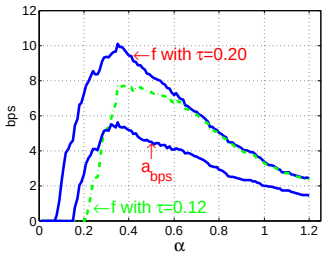


Fig. 11. f and a_{bps} with respect to α on one dataset.

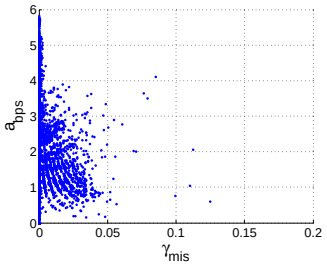


Fig. 12. Approximate bit rate a_{bps} and its mismatch rate γ_{mis}.

The message exchanging mentioned above does not compromise the security level. First, it is almost impossible to predict the fluctuation of the residual readings from exposed historical data segments nearby because of the short coherence time. Thus, the adversary cannot learn more about the final secret bits generated from unrevealed residual data segments. Second, disclosed parameters after training only reflect the statistics of the noise. This information may indicate the secret bit rate of our scheme in a short time period, but this is fine since the secret bit rate is publicly known.

VII. Experiment

We evaluate the proposed methods based on measurements from real vehicular networks. To collect the measurements, the two cars, Alice and Bob, drive one after the other in the same direction on the same route at the same time. Both of them are carrying laptops running Ubuntu 9.10 with Atheros WiFi chipsets and external antennas mounted on top of the cars. While driving Alice keeps sending probes, which only have a preamble and an index payload, to Bob, and Bob immediately returns back corresponding replies upon reception. Both sides record, as datasets, the RSSI readings of received packets along with their indices. We conduct measurements for 8 runs over two different routes, generating 8 datasets. Datasets $1 \sim 5$ are from a suburban route with rate of 500 probes per second, while datasets $6 \sim 8$ are from a rural route with 1000 probes per second. The two routes are shown in Figure 13. It is worth mentioning that these probing rates are for the establishment of the secret key, which takes only a few seconds. In the following, we first show how to optimize some system parameters, and then apply them into our online procedure to show how fast we can extract bits. Finally we validate the randomness of the final bits by the NIST test suite.

A. Parameters optimization

We take a_{bps} as the metric for the system parameter optimization. As shown in Figure 12, maximizing a_{bps} does not necessarily increase mismatch rate, thus it is safe to optimize a_{bps} without concern of the mismatch rate. This single metric not only simplifies the optimization procedure, but also inspires our design of online objective function f.

Picking m and s Figure 14 shows a_{bps} with respect to three parameters for one dataset. According to the figure, we

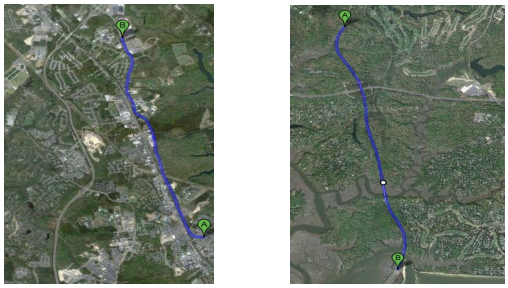


Fig. 13. Routes for data collection: suburban route (left), and rural route (right). (GoogleEarth)

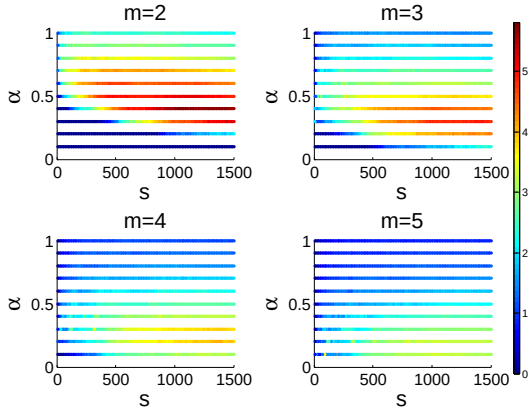


Fig. 14. Influence of three parameters. Objective a_{bps} is represented by color. We choose $m = 2$ and s as 2-second readings.

choose excursion threshold $m = 2$ because of consistently better performance. The sliding window size s does not influence a_{bps} much, especially after $s = 1000$. Thus we set it to be 2 seconds long because the probing rate of this dataset is 500 per second. However, a_{bps} changes rapidly with respect to α, so we do not fix it here and leave it to be optimized.

Data smoothing We smooth the data to reduce noise, which should not be confused with the sliding window smoothing in level crossing to subtract slow variation. We implement three techniques: (1) k-reading average; (2) sliding window smoothing; (3) CCA-based smoothing. Figure 15 shows the result. k-reading average hurts the performance. Sliding window smoothing and CCA-based smoothing can improve the performance before k=3. For k>3, performance is worse because some information is smoothed out. CCA-based smoothing outperforms sliding window smoothing consistently. To study why CCA smoothing performs the best, we look into its weights at $k = 3$ and find that $\mathbf{a}$ =(0.40,0.36,0.24), $\mathbf{b}$ =(0.23, 0.36, 0.41). Note that $\mathbf{a}$ and $\mathbf{b}$ are nearly the inverse of each other, which can compensate for mismatched sensing time. Thus, the benefit of CCA comes from both smoothed local noise and adjusted sensing time. Consequently, it can generate 50% more entropy bits compared to raw readings.

Online scheme There are two parameters for online study: training ratio γ_r and frame length $\overline{f}$. We compute for $\gamma_r = 0.10, 0.12, \ldots, 0.60$ and $\overline{f} = 1, 2, \ldots, 20$ (e.g., $\overline{f} = 1$ means 1 second readings). Figure 16(a) shows that single-frame scheme does not have a generalizable parameter setting. The double-frame scheme improves it significantly

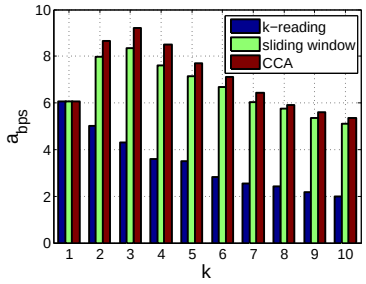
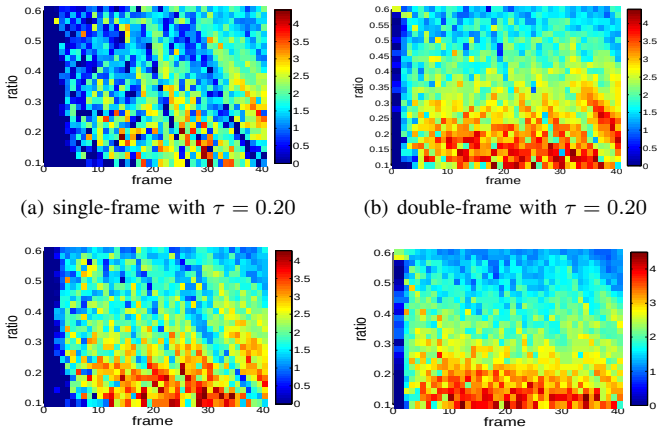


Fig. 15. Smoothing techniques. When $k = 1$, no smoothing is performed. We choose CCA with $k = 3$.

(Figure 16(b)). Therefore, overestimation indeed hurts less than underestimation. In sight of this, we set $\tau = 0.12$ which gives higher α than $\tau = 0.20$, and the result is in Figure 16(c). We can further improve it via the double-frame technique (Figure 16(d)). Observe that $\gamma_r = 0.10 \sim 0.16$ and $\overline{f} = 10 \sim 20$ perform equally well. Therefore, we set $\gamma_r = 0.10$ and $\overline{f} = 10$.



(a) single-frame with $\tau = 0.20$ (b) double-frame with $\tau = 0.20$

(c) single-frame with $\tau = 0.12$ (d) double-frame with $\tau = 0.12$

Fig. 16. a_{bps} vs. ratio (γ_r) and frame ($\overline{f}$). The value of a_{bps} is represented by color. We choose double-frame scheme with $\tau = 0.12, \overline{f} = 10, \gamma_r = 0.1$.

B. Experimental results

We compute the entropy bit rate for all datasets. We use two schemes, an offline scheme and an online scheme. For the offline scheme, the weights for CCA are picked offline, and α is chosen via Algorithm 2. For the online scheme, we use the double-frame scheme with CCA incorporated, the weights of CCA and α are learned online. Table I shows the results. First, there is no obvious correlation between driving speed and a_{bps}. The trace contains sufficient variations. The main factor limiting a_{bps} is the noise contained in the readings. Second, the offline scheme can get 6-9 entropy bits per second for datasets 1-6, while it performs poor on datasets 7 and 8. The online scheme offers more consistent performance. It can get 4-9 entropy bits per second for all datasets. This is because, the offline scheme uses a single α and there is always some bad portion of readings in some datasets that cause many mismatches. On the other hand, the online scheme, discards some data for training, adaptively adjusts parameters and achieves better performance. Thus, the online scheme

TABLE I. EXPERIMENT RESULTS. CAR SPEED IS IN MPH.

Data		Offline			Online		
no.	speed	a_{bps}	s_{bps}	γ_{mis}	a_{bps}	s_{bps}	γ_{mis}
1	25	7.56	3.98	0.0000	5.45	3.28	0.0000
2	35	6.46	2.97	0.0000	4.53	2.66	0.0010
3	35	7.95	4.54	0.0011	5.63	3.33	0.0000
4	45	9.23	4.96	0.0000	5.62	3.42	0.0000
5	45	7.78	4.31	0.0000	5.16	2.92	0.0000
6	35	7.81	0.00	0.0004	8.86	3.90	0.0000
7	45	1.11	0.00	0.0000	4.40	1.74	0.0000
8	50	3.34	0.00	0.0000	5.08	1.75	0.0000

offers more consistent performance in different environments. Third, s_{bps} is smaller than a_{bps}. This is caused by the efficiency of the randomness extractor. Our current implementation uses $N = 20$ in the extractor. In principle, we can get s_{bps} arbitrarily close to a_{bps} by increasing N in the extractor. But there are computational challenges, so we reserve this for future work. For the three cases where $s_{bps} = 0$, this is caused by efficiency: $s_{bps} > 0 \Longleftrightarrow \eta > \frac{n_p}{n_q} E$.

C. Proof of security

In vehicular environments, the adversary, Eve, is at least a few meters away from both Alice and Bob. Due to the nature of wireless fading, Eve cannot get any information about the dynamics experienced by Alice and Bob, which has been verified [5], [8], [15]. In addition, the information exposed by information reconciliation has been removed by privacy amplification. Thus, the only security concern left is whether the extracted bits are sufficiently random. To validate randomness, for each s_{bps} in Table I, we perform the NIST tests on the extracted bits. Note that even for the traces where $s_{bps} = 0$, we obtain extracted bits. The zero bit rate is caused by subtraction of leaked information. Among the 15 tests in the NIST tool, we run 8 tests and find that the extracted bits can pass these tests. The other tests require large input size, and we plan to run them in the future.

VIII. CONCLUSION

In this work, we consider the secret key extraction problem in vehicular networks. Our solution improves on an existing level crossing technique to work in a noisy vehicular environment. Measurements from real world vehicular networks show that we can extract 6-9 bits per second in most traces, and adjusting parameters in real-time can help tolerate noise in different environments and offer steady performance.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-1117412, CAREER Award CNS-0747108, CNS-1156574, China NSF grants (61073152, 61133006) and China 973 project (2012CB316200).

REFERENCES

[1] W. Viriyasitavat, F. Bai, and O. K. Tonguz, "Dynamics of network connectivity in urban vehicular networks," *IEEE Journal on Selected Areas in Communications*, 2011.

[2] J. J. Haas, Y.-C. Hu, and K. P. Laberteaux, "The impact of key assignment on vanet privacy." *Security and Communication Networks*, 2010.

[3] I. Rouf, R. D. Miller, H. A. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study." in *USENIX Security Symposium'10*.

[4] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure vehicular communication systems: design and architecture," *IEEE Communications Magazine*, 2008.

[5] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, "Radio-telepathy: extracting a secret key from an unauthenticated wireless channel," in *MobiCom '08*.

[6] C. Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N. B. Mandayam, "Information-theoretically secret key generation for fading wireless channels," *IEEE Trans on Information Forensics and Security*, 2010.

[7] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in real environments," in *MobiCom '09*.

[8] N. Patwari, J. Croft, S. Jana, and S. K. Kasera, "High-rate uncorrelated bit extraction for shared secret key generation from channel measurements," *IEEE Trans on Mobile Computing*, 2010.

[9] L. Cheng, B. E. Henty, F. Bai, and D. D. Stancil, "Doppler spread and coherence time of rural and highway vehicle-to-vehicle channels at 5.9 ghz," in *GLOBECOM'08*.

[10] J. Camp and E. Knightly, "Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation," in *MobiCom '08*.

[11] NIST, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," 2001.

[12] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener, "Robust key generation from signal envelopes in wireless networks," in *CCS'07*.

[13] H. Liu, J. Yang, Y. Wang, and Y. Chen, "Collaborative secret key extraction leveraging received signal strength in mobile wireless networks," in *INFOCOM'12*.

[14] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li, "Imdguard: Securing implantable medical devices with the external wearable guardian," in *INFOCOM'11*.

[15] M. Wilhelm, I. Martinovic, and J. B. Schmitt, "Secret keys from entangled sensor motes: implementation and analysis," in *WiSec '10*.

[16] Z. Yang, A. C. Champion, B. Gu, X. Bai, and D. Xuan, "Link-layer protection in 802.11i wlans with dummy authentication," in *WiSec'09*.

[17] H. Wang, B. Sheng, and Q. Li, "Privacy-aware routing in sensor networks," *Elsevier Computer Networks*, 2009.

[18] H. Wang, B. Sheng, C. Tan, and Q. Li, "Wm-ecc: An elliptic curve cryptography suite on sensor motes," *College of William and Mary, Computer Science, Williamsburg, VA, Tech. Rep. WM-CS-2007-11*, 2007.

[19] F. Liu, X. Cheng, L. Ma, and K. Xing, "Sbk: a self-configuring framework for bootstrapping keys in sensor networks," *IEEE Transactions on Mobile Computing*, 2008.

[20] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in *EUROCRYPT '93*.

[21] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," *SIAM J. Comput.*, 1988.

[22] D. R. Hardoon, S. R. Szedmak, and J. R. Shawe-taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Comput.*, 2004.

[23] A. V. Prokhorov, "Paritial correlation coefficient," *Encylcopaedia of Mathematics*, 2001.

[24] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, 1978.

[25] P. Elias, "The efficient construction of an unbiased random sequence," *The Annals of Mathematical Statistics*, 1972.

[26] R. Shaltiel, "Recent developments in explicit constructions of extractors," *Bulletin of the EATCS*, 2002.

[27] M. Blum, "Independent unbiased coin flips from a correlated biased source: a finite state markov chain," *Combinatorica*, 1986.

[28] J. Kim, "Iterated grid search algorithm on unimodal criteria," Ph.D. dissertation, 1997.