Side-Channel Security Analysis of Connected Vehicle Communications Using Hidden Markov Models

Fei Sun, Richard R Brooks, *Senior Member, IEEE*, Gurcan Comert, and Nathan Tusing

In intelligent transportation systems (ITS) applications, dedicated short-range communication (DSRC) applications and network stacks are not yet mature. They have not been adequately tested and verified. This paper investigates side-channel vulnerabilities of a wireless communication application in vehicular environments (DSRC/WAVE) protocol implementation of a traffic intersection application. A prototype roadside unit (RSU) was implemented using real DSRC devices. The functionality of the WAVE short message (Wsm)-channel is extended to include an implementation of WAVE short message protocol (WSMP) for broadcasting GPS data and RSU instructions in vehicular communications. In the example used, DSRC is used to replace an intersection stoplight. Denial of service attacks are executed that leverage DSRC RSU timing and packet size side-channels to selectively disable the stoplight. Simulations are implemented to determine our ability to stealthily drop packets so as to force two vehicles to collide. Hidden Markov models (HMM) and Support Vector Machines (SVM) are constructed from sniffed side-channel information. We use inter-packet delay time and packet size side-channel information to design our attackes. In operational networks, packets should be encrypted in order to hide the contents of the packet payloads, but packet sizes and timing are not affected by encryption. HMMs were inferred using only side-channel information. The inferred HMMs track the protocol status over time. The SVM classifier was inferred using both side-channel data and packet payloads. At run-time, though, the SVM only had access to side-channel information. Simulation experiments were implemented to test HMM and SVM ability to identify packets used to signal automobiles to stop and yield right-of-way. The simulations showed that for HMMs the timing side-channel attack was most effective, dropping the packets needed to cause a collision with only a 2.5% false positive rate (FPR), while the packet size side-channel attack works with 9.5% FPR. The SVM classifier using both side-channels had a higher true positive rate (TPR) of 72.5%, but also had a higher FPR 20%. In contrast, dropping packets at random had both high FPR and TPR.

Index Terms—**Dedicated short range communications, hidden Markov models, side-channel analysis, cyber security, connected and autonomous vehicles, traffic intersection.**

I. Introduction

DEDICATED Short Range Communication (DSRC) is 802.11p based wireless communication technology widely used for communication between vehicles and the surrounding infrastructure. Wireless access in vehicular environments (WAVE) is one of the communication protocols of DSRC. It provides stable, high-speed communication between connected vehicles. Many applications based on DSRC/WAVE are being developed to improve traffic efficiency and assist driving and vehicle-to-vehicle (V2V) technology. Vehicles use V2V and a global positioning system (GPS) to share and detect information within range, e.g., to alert and warn drivers for conflicts that may not be easy to see or perceive in time. In left turn assist (LTA), the system help avoid blind spots when drivers turn left, and it warns drivers if they are driving in front of another vehicle traveling in the opposite direction [1], [2].

With DSRC becoming the vehicle-to-everything (V2X) wireless mobility standard, DSRC protocols, applications, and stacks need to mature. Likewise, an immense number of potential applications using the DSRC protocol have not been adequately tested and verified [3]–[6]. In this study, a side-channel (*black box*) analysis of WAVE short message protocol (WSMP), the messaging protocol used by DSRC/WAVE, is carried out using Hidden Markov models. It is assumed that the WSMP packets are encrypted in the side-channel analysis. Thus, the methodology does not depend on the shared contents (*e.g.*, basic safety messages (BSMs)).

The experiments are conducted on Cohda fifth-generation On-Board Unit (OBU) equipment. A simplified connected traffic control system is simulated under the connected and autonomous vehicles framework to implement the security analysis. The infrastructure system runs on a roadside unit denoted as RSU, a smart roadside DSRC unit with computational capabilities for traffic control. The application works to avoid crashes for connected and autonomous traffic.

The side-channel analysis is conducted by the sniffed WSMP traffic. Hidden Markov Models (HMMs) are built using sniffed packet traces, a Support Vector Machine classifier is built to classify critical packets based on the HMM outputs. Then critical packets in the system are identified and predicted using the HMMs. A critical packet refers to the stop instruction packet sent from a roadside unit (RSU) to tell a vehicle to stop to avoid a collision. Targeted attacks exploit this known weak point. To test our approach, we created traffic simulations that triggered the DSRC application. HMMs were used to analyze the DSRC data and detect when the next packet to be sent would be the one that told a vehicle to stop. We refer to these packets as *critical* packets, since these are the packets that are used to protect vehicles and passengers.

This study was based upon a project funded by the US Department of Transportation (USDOT) Center for Connected Multimodal Mobility (C2M2) (Tier 1 University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, USA.

Manuscript received December, 2020; Corresponding author: G. Comert (email: gurcan.comert@benedict.edu). R. R. Brooks is with the Holcombe Department of Electrical and Computer Engineering, Clemson, SC 29634, USA. F. Sun received her M.S. degree from the Holcombe Department of Electrical and Computer Engineering, Clemson, SC 29634, USA. She is now with Sichuan Suitang Tech. Co. Ltd., Chengdu, Sichuan, China. G. Comert is with the Department of Computer Science, Physics, and Engineering, Benedict College, Columbia, SC 29204, USA. Nathan Tusing is with the Holcombe Department of Electrical and Computer Engineering, Clemson, SC 29634, USA.

The denial of service (DoS) attack simulation uses HMM and SVM predictions in simulation experiments where packets are dropped to provoke vehicle crashes in the simulations. Packet flooding attacks were performed instead of wireless jamming because (1) Federal Communications Commission (FCC) regulations make research use of wireless frequencies for jamming problematic, (2) packet flooding tools are widely available, well understood, and easy to use [7], and (3) where wireless jamming requires physical proximity, packet flooding attacks can be launched from anywhere on Earth. Packet flooding attacks on DSRC applications are therefore simpler and potentially more problematic attacks. That said, our side-channel inferences would be equally useful for triggering wireless signal jamming.

A. Background

In this subsection, relevant background information is provided on DSRC security and the methods used to build HMMs and SVMs. This reviews the methods used for protocol analysis, side-channel analysis of the traffic control system, building, and testing of HMMs/SVMs.

1) DSRC

DSRC communications technology was developed for connected vehicle applications [8] in order to meet safety-critical application requirements (e.g., latency and packet drop rates). DSRC provides reliable and real-time communication between DSRC-equipped vehicles. It has started to be widely used for daily traffic operations to move people and goods from emergency vehicle preemption to pedestrian crossings. In twenty years, the U.S. Department of Transportation (USDOT) report [9] predicted that most lights and traffic signals would enable DSRC. It is expected to have DSRC or similar connectivity as a majority of the crashes would be avoided by some way of alerts. Connected vehicles (CVs) can share critical information, so it provides the possibility of unobstructed awareness.

In this study, we use the architecture of the DSRC standard from [10]. The physical (PHY) protocol, including PHY layer and medium access control (MAC) sublayer, is defined in IEEE 802.11p wireless access in vehicular environments (WAVE), which enhances the IEEE 802.11 (WIFI standard) to support Intelligent Transportation System (ITS). It provides real-time data exchange by removing the need to establish channels. WAVE defines the DSRC channels in the U.S. The authentication and data confidentiality mechanisms provided by the IEEE 802.11 standard cannot be used. DSRC equipped vehicles within line-of-sight can receive data frames as soon as they arrive.

2) DoS Attack on DSRC

In order to disrupt the latency and increase the packet drop rates in a safety-critical application (i.e., intersection control), we perform denial of service (DoS) attacks on DSRC channels. DoS network attacks hamper access by legitimate users to a network service by consuming network or CPU resources. There are several methods for performing DoS attacks. Packet flooding consumes network bandwidth. Amplification uses services where response frames are much larger than requests (e.g., domain name system (DNS) requests). SYN flooding attacks utilize the time-out mechanism of transmission control protocol (TCP) sessions [7], [11]–[13].

In this study, since DSRC uses channels without session set-up, packet flooding is used to occupy DSRC channel bandwidth, resulting in legitimate packet dropping [5], [6]. Other researchers have investigated DSRC DoS attacks [14]–[17]. Laurendeau et al. [18] found DoS to be the major risk in their DSRC threats analysis. The paper also pointed out that DSRC standards should enhance the security of the lowest possible layer to prevent DoS, such as providing link layer authentication. Islam et al. [19] developed an application CVGuard for DoS attack detection and prevention. The application was designed to monitor the context of DSRC communication and detect the attack based on road policies and rules. As a side-channel analysis, only the properties of a higher level of the data exchanged are investigated in this paper.

3) Models Used

We used three different approaches for driving proof of concept DSRC DoS attacks:

Random packet dropping is used as a simple approach for comparison with the other two models. In this approach, flooding traffic is inserted into the system at random intervals. The frequency of the attack is changed to vary the percent of packets that would be dropped by the system. Packet dropping rate is varied from 10% to 90%. No attempt is made to identify which packets are most critical to system performance. The effectiveness of random packet dropping for causing collisions between vehicles can be found in Table VIII.

Hidden Markov models (HMMs) can be used to model dynamic systems that include a stochastic component. The HMM approach used originated with Shalizi et al.'s [20] causal state splitting and reconstruction (CSSR) algorithm. CSSR generates HMMs directly from discrete data sequences. It infers the model structure (the number of hidden states and their transition structure) from the observation sequence and a parameter (L) that defines how many time steps could influence state transitions and indirectly the size of the state-space. HMMs derived using CSSR have predictive best explain the system's stochastic components. Schwier et al. [21] extended CSSR to create zero-knowledge HMM inference for automatically inferring the state-space of the HMM from the training data as part of model construction. Typical HMM's have two sets of probability density functions (pdf) associated with the state-space: one pdf for state transitions and another hidden pdf for observation production. In the CSSR approach, state transitions generate observations. This simplifies the underlying model by requiring only one pdf. Without loss of information, it is possible to generate one pdf HMMs from HMMs with two pdfs [20]. The zero-knowledge method infers the HMMs using only observation sequences. The algorithm assumes only that the system has an underlying state-space and that probability distributions are stable, i.e., that a Markov model can express the underlying system. This approach evaluates the training data used to learn the model to see if it can provide the model certainty desired [22].

We use HMMs for protocol inference, since we have successfully used them for similar applications in the past. This includes NATO tracking maritime traffic [21], identifying languages typed within encrypted streams [23], and denial of service for power networks [24]. Harakrishnan et al. [23] proposed timing side-channel analysis for detecting protocol tunneling. They used the zero-knowledge approach [21] to extract HMMs for extracted keystroke dynamics of languages. They then used the HMM for language detection. Zhong et al. [24] proposed the side-channel analysis of the Phasor Measurement Unit (PMU) protocol used by the communications network of the smart grid. They isolated the packets of the target PMU sent through a VPN channel shared with other PMUs, followed Denial-of-Service (DoS) attacks that selectively drop packets from the target PMU. Similarly, the authors in [25] used HMM to predict user activity to assign channels for secure data transmission to improve efficiency and security.

Note that network protocols are typically designed with an underlying state-space. At each state, the system performs processing to determine its response. The amount of computation required within a state would vary little but vary greatly between states. The packet size of a given response (ex. ACK) would be relatively constant, but the packet sizes for different classes of responses would vary greatly. This makes HMM inference a natural tool for analyzing protocol side channels.

Support Vector Machines (SVM) using a radial basis function kernel are trained to identify packets that immediately precede packets containing **stop** messages. When the next packet is expected to tell a vehicle to stop, a DDoS action is unleashed to keep that packet from arriving at its destination. This frequently results in collisions in the intersection. Note that HMMs have states that need to be identified, but SVMs do not. This forces us to identify classes of packets instead of model states. SVMs separate clusters of data using hyperplanes. Packets are identified by observing two features: packet size and inter-packet delay (see Fig. 11). These are the same features used by the HMMs, and the reason for using those features is the same reason HMMs use them. These features are not affected by most current protocol security approaches. Since the HMM and SVM approaches use the same data sources and trigger the same attack events, we can directly compare the true and false positive rates of these two approaches.

Support Vector Machines were proposed by Cortes and Vapnik in 1995 [26]. By using labeled data from feature space R^n (i.e. there are n features), the best possible separating hyperplane can be found starting with the Eq. (1):

$$y_i(wx_i + b) \geq 1, i = 1, \ldots, m \tag{1}$$

where y_i is the class associated with the features x_i, m is the number of points in the training set, b is a constant, and w is a vector that scales the x_i to fit the constraint. The goal of this problem is to find w_0 where

$$w_0\mathbf{x} + b_0 = 0 \tag{2}$$

Eq. (2) represents the optimal separating hyperplane with maximum margins between the two linearly separated classes.

SVMs have been used in multiple areas, including different traffic analysis type problems. These include traffic prediction [27][28], flow classification [29], and DDoS detection [30]. Our usage of SVM does not differ from the original formulation of SVM, but it presents the novelty of predicting critical packets from network side-channel data alone.

4) Vehicle simulation

For numerical evaluation, vehicle movements are simulated in Python, but their communications are sent through real DSRC equipment and OBUs. The application Wsm-channel running on two real OBUs and generating real DSRC communication traffic is used. Furthermore, the communication traffic contents between two OBUs are simulated vehicle movements. After collecting communication traffic on real DSRCs, HMMs are built. Then, an attack simulation is conducted offline, which is not on real DSRCs.

The paper is organized as follows. Section II describes the communication protocol, traffic intersection simulation, side-channel analysis framework, and HMM learning algorithms. Section III presents our data generation process, HMM and SVM models inferred using side-channel analysis and experiment evaluations. Finally, section IV summarizes findings and possible future research directions.

II. Methodology

A. *Simple Connected Intersection Control Simulation*

As for connected and autonomous vehicles, ITS application, a connected traffic intersection simulation is developed for experiments. Given in Fig. 1, the traffic intersection control is designed for two-way 2-lane roadways. Vehicles are generated from Uniform interarrivals. The designed intersection is controlled under the following assumptions:

1) Pedestrians are not allowed or considered.
2) Vehicles can come from one of the four directions North (N), South (S), West (W), or East (E), and go straight, left, or right. U-turn is not permitted at this crossroad.
3) All vehicles are connected autonomous vehicles (CAVs) and controlled by onboard unit (OBU) speed control. There is a central RSU (i.e., roadside unit-a smart roadside unit with computational capabilities) in the center of the intersection. They change speeds only following the instruction from RSU.
4) The center of the intersection is set to (0, 0) as the origin O.
5) All lanes are 4 meters wide.
6) As shown in Fig. 1, the exit points of each lane are A, B, C, and D. Coordinates are A (-4, -2), B (4, 2), C (-2, 4), D (2, -4). Vehicles would start to report information 50 meters away from the RSU (O in Fig. 1).
7) As shown in Fig. 1, the entry points are A', B', C', and D'. Coordinates are A' (-4, 2), B (4, -2), C (2, 4), D (-2, -4).
8) The path of the vehicle in the intersection is calculated by linear distance from exit points to entry points. The distance a vehicle should drive in the intersection is calculated by the sum of path distance and vehicle length. For example, if a vehicle is driving from East

to North, the path in the intersection should be line BC' (Fig. 1).

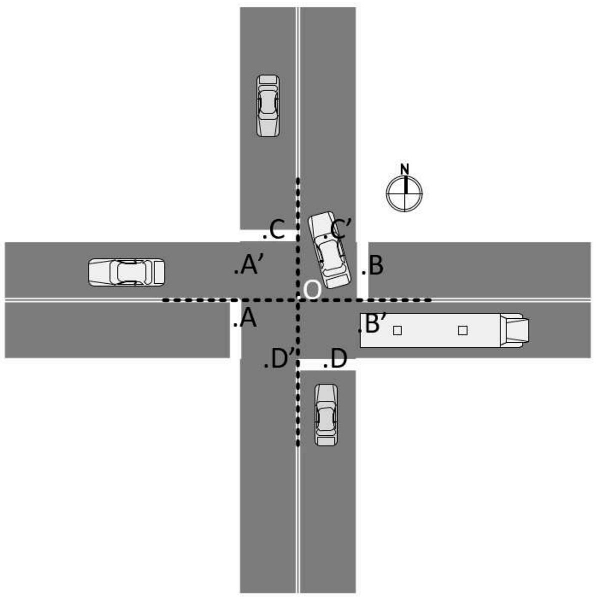


Fig. 1. Simulated simple intersection control

Each vehicle sends its information to RSU when it is 50 $meters$ away from the intersection. The RSU estimates vehicles' arrival time at the intersection and sends stop instructions if the intersection is busy at the estimated arrival time. A *stop instruction* includes the vehicle's information and its timing to drive into the intersection.

B. Communication Protocol Implementation

This study tests a reliable WAVE short message protocol (WSMP) communication application Wsm-channel. WSMP, IEEE 1609.3, is a DSRC based communication protocol that allows data rates parameters [10]. Wsm-channel could broadcast GSP information of host OBU on a Wsm-channel. In this study, the forward transmit (FWDTX) and forward receive (FWDRX) on Wsm-channel modes are implemented to forward packets through different protocols. FWDTX is forwarding received UDP packets to the WAVE protocol. FWDRX is forwarding received WAVE packets to the UDP protocol. Thus, using this extended application, processes on different OBUs can exchange data.

The flowchart of communication is given in Fig. 2. For example, if Process A on DSRC1 needs to send packet A to Process I on DSRC2; Process II receives packet A and sends packet B back to Process I. Wsm-channel FWDTX and FWDRX modes are running on DSRC1 and DSRC2. Process I and Process II are listening to UDP for receiving packets. The communication steps are as follows:

1a. DSRC1: Process I sends packet A to UDP.
1b. DSRC1: Wsm-Channel FWDTX thread receives packet A and sends it to WSMP at interface wave-raw. Packet A is broadcasting at wave-raw.
2a. DSRC2: Wsm-channel FWDRX thread receives packet A at wave-raw and sends it to UDP.
2b. DSRC2: Process II receives packet A.
3a. DSRC2: Process II generates packet B and sends it to UDP.
3b. DSRC2: Wsm-channel FWDTX thread receives packet B and sends it to WSMP at interface wave-raw. Packet B is broadcasting at wave-raw.
4a. DSRC1: Wsm-channel FWDRX thread receives packet B at wave-raw and sends it to UDP.
4b. OBU1: Process I receives packet B.

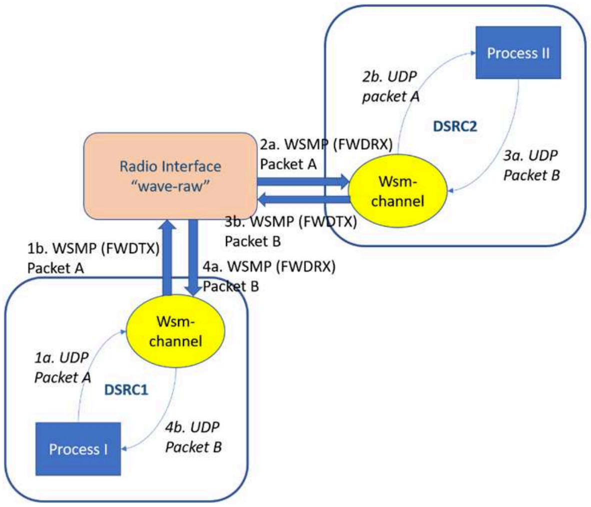

Process II
2b. UDP packet A
DSRC2
3a. UDP Packet B
Radio Interface "wave-raw"
2a. WSMP (FWDRX) Packet A
Wsm-channel
3b. WSMP (FWDTX) Packet B
1b. WSMP (FWDTX) Packet A
4a. WSMP (FWDRX) Packet B
Wsm-channel
1a. UDP Packet A
DSRC1
4b. UDP Packet B
Process I

Fig. 2. Flowchart of a packet within DSRC communication

In the simulation, communications between an onboard unit (OBU) and a roadside unit (RSU) are focused. The OBU stores the information of all the cars approaching the intersection, and the RSU serves as the roadside or control unit for the intersection. Thus, communications between cars and RSU over the DSRC channel, wave-raw, are observed as illustrated in Fig. 3.

In this study, the two DSRCs (OBU and RSU) are real equipment. The radio interface is the communication channel, Wsm-channel is the application used for communication between two real DSRC equipment. A Python script of simulated vehicle movements is Process I in DSRC1, and the controller script in Python is Process II in DSRC2.

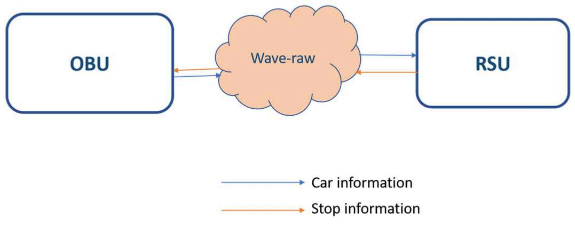


Fig. 3. Communication between OBU and RSU

C. Side-Channel Analysis

As the IEEE1609.2 standard of the DSRC/WAVE stack defines the standard mechanisms for authenticating and encrypting messages, the 'black box' analysis is considered. The side-channel characteristics (packet size (in Bytes (B)), packet inter-delay in seconds (s)) of WAVE short message protocol (WSMP) are looked at. Even if encryption and authentication are implemented as specified in the IEEE 1609.2 standard, DSRC/WAVE may still be susceptible to such black box analysis that does not depend on the contents.

According to the sniffed traffic (Table I), where the time refers to the inter-packet time (s), the packets are not arriving at the same rate all the time, which means the protocol is not active all the time. Packet sizes are shown in Bytes (B). If the attack is performed at an inactive time, one cannot cause any trouble. Moreover, since flooding traffic is easy to recognize, the devices may lose access to the channel.

TABLE I. Sniffed DSRC traffic example

No	Time (s)	Source	Protocol	Size (B)	Info.
1	0.000000	b8:ff:36:ff:36...	WSMP	176	WAVE S. Mes. Pro. IEEE P1609.3
2	0.004006	b8:ff:36:ff:36...	WSMP	178	WAVE S. Mes. Pro. IEEE P1609.3
3	0.270763	b8:ff:36:ff:36...	WSMP	177	WAVE S. Mes. Pro. IEEE P1609.3
4	0.025839	a8:ff:36:ff:36...	WSMP	234	WAVE S. Mes. Pro. IEEE P1609.3
5	6.853376	b8:ff:36:ff:36...	WSMP	177	WAVE S. Mes. Pro. IEEE P1609.3
6	0.027991	a6:ff:36:ff:36...	WSMP	233	WAVE S. Mes. Pro. IEEE P1609.3
7	0.212516	b8:ff:36:ff:36...	WSMP	177	WAVE S. Mes. Pro. IEEE P1609.3
8	0.230590	ba:ff:36:ff:3c...	WSMP	177	WAVE S. Mes. Pro. IEEE P1609.3
9	0.030930	a8:ff:30:ff:36...	WSMP	234	WAVE S. Mes. Pro. IEEE P1609.3
10	6.368706	b8:ff:36:ff:36...	WSMP	193	WAVE S. Mes. Pro. IEEE P1609.3

Two network protocol analysis methods based on side-channels: Hidden Markov model (HMM) and support vector machine (SVM) are developed. The process flow for HMM inference is in Algorithm 1. Protocol analysis HMMs help reverse engineer the protocol. The SVM is a common machine learning problem that is used for data classification. HMMs provide a model of the observable network process. SVMs provide a tool for separating the network protocol packets into classes. In this work, both are used to identify packets that should be dropped in order to cause the DSRC traffic control algorithm to fail and trigger crashes. Once HMM is used to discover the structure of the protocol and which packets to remove from the system, it is discovered that SVM can better identify those packets and more effectively disrupt the system. The HMM inference is explained in more detail here since SVM learning is already well established.

As WAVE packets are assumed to be encrypted, size and timing side-channels are applied. Traces of DSRC network protocols are sniffed. Identify network protocol states that can be identified by using observed packet characteristics to associate each sniffed packet with a class. Protocol participants are known. Their positions in the sequence give transitions between protocol states. With the HMM, target packets of stop information sent by RSU are successfully isolated, following DoS attacks that selectively drop packets from RSU. The goal is to side-channel vulnerabilities of WAVE protocol assuming all the security services are implemented.

1) Side Channel Symbolization

This Section explains how to infer sets of observations/events to use for HMM and/or SVM analysis. Inter-packet timing and packet size are two commonly used features for network protocol side-channel analysis [24]. For both features, histograms are plotted of observation value frequencies. Distinct peaks in the histograms can be differentiated, and each peak becomes a distinct observation symbol. In practice,

Algorithm 1 Inferring HMM [31]

```
1:  Observe data
2:  Symbolize
3:  for $i$ in 2:$N$ do
4:      Infer transition matrix $L = i$
5:      if Enough data then Construct states
6:          if Any identical states then Merge states
7:          else Construct HMM
8:              if States converged then Model built
9:              else $i$=$i + 1$
10:             end if
11:         end if
12:     else Observe more data
13:     end if
14: end for
```

this approach has been generalized for an arbitrary number of dimensions, see [32]. In this work for simplicity, these two sets of features are treated independently, but a two dimensional vector of features would also be possible. The two sets of features are used together in the SVM. Raw inter-packet timings and packet sizes are fed directly into the classifier.

From the sniffed traffic (see Table I), the side-channels: timing and size are analyzed. Inter-packet delay is used instead of latency for analysis. Inter-packet delay, also known as delta time, is calculated by subtracting the previous packet's receive time from the current packet's time. One can start with $i = 2$ (*e.g.* and inter-packet delay $\Delta t_{i=2} = (t_2 - t_1)$ seconds). Inter-packet delay is a more robust side-channel feature since it depends mainly on the network protocol and not the network topology.

First, a histogram of inter-packet delays is plotted, shown in Fig. 4. Note that there are two distinct peaks. Any packet whose delta-time is within a given peak is assigned the same symbolic value, shown in Table II. This translates the data time series into a long string of symbols. The same process is followed with packet size pattern values (see Table III). These strings are used to infer HMM.

TABLE II. Timing observation ranges

Observation type	Timing range (s)
A	$\Delta t <$ 0.06
B	0.06 $\leq \Delta t \leq$ 1.00
C	$\Delta t >$ 1.00

TABLE III. Size observation ranges

Observation type	Size range (B)
x	$s \leq 210$
y	$s > 210$

D. Hidden Markov Model Inference

The HMM is used to analyze side-channel information. In this process, there are no *a priori* states. With state-

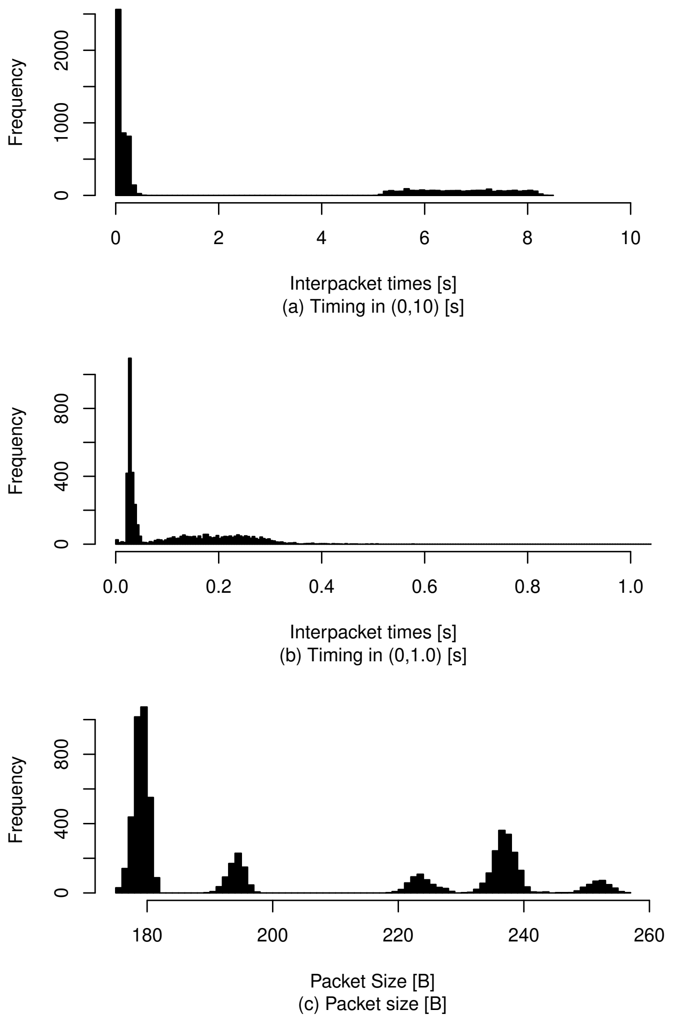


Fig. 4. Timing and packet size histograms

space parameter $L = 1$ states are associated with observations. When a transition occurs using output symbol a_i, the system moves to state a_i. For state-space parameter $L = n$, when a transition occurs using output symbol $a_{i(n+1)}$, the system moves from current state $a_{i1}a_{i2},...,a_{in}$ to state $a_{i2}a_{i3},...,a_{in}a_{i(n+1)}$. Algorithm 2 shows how state transition probabilities are inferred by considering the set of observations as a time series generated by the process to be modeled. The conditional probability of symbol a_i following symbol a_j (transition probability from state i to state j) is determined by frequency counting. The state merging algorithm (Algorithm 3) finds equivalent states and merges them into a single state. This extends the approach in [33] by applying the z-test in Algorithm 4 to the HMM and rigorously determining the statistical significance of the inferred model. Unlike traditional HMM approaches, the approach in this study can establish that the training data sample is adequate for providing the data certainty needed [31]. Pearson chi-square test proves the significance of evidence to merge two similar states [34]. The confidence interval approach provides the level of acceptance for putting a string into an HMM [35].

A standard HMM has two sets of random processes, one governing state transition and the other governing symbol outputs. In this paper, the representation of an HMM in [31] is used, where output symbols are associated with transitions (see Algorithm 2). The two approaches are equivalent [21]. This representation uses a tuple G=$\langle A, V, E, P\rangle$, where A is a finite alphabet of observations, V is a finite set of nodes or states, $E \subseteq V \times A \times V$ is a transition relation, and $P : E \rightarrow [0, 1]$ is a probability function such that $\sum_{a\in A, v_j inV} p(v_i, a, v_j) = 1$. Each element $p_{i,j} \in P$ expresses the probability the process transitions to state v_j once it is in state v_i. For each pair of (v_i, v_j), $E(v_i, v_j) = a_i$. It should also meet the requirement that if $E(v_i, v_j) = a_i$, then $E(v_i, v_j) \neq a$, where $v_i, v_j, v_k \in V$.

Both state transition probability matrix P and state output probability matrix O can be constructed from G. The state output probability matrix refers to the matrix that describes the probability distribution of the next observation for each state. The state transition probability matrix is used for steady-state probability calculation and plotting figures. The state output probability matrix is used for generating a string from the HMM and HMM acceptance checking. Following are some important variable calculations in an HMM.

i. Conditional probabilities which are denoted as $p_{i,j} = p(v_j|v_i)$.
ii. Transition count $c_{i,j}$ is the number of transitions from i to j happened.
iii. Count $c_i = \sum_j c_{i,j}$=number of state i is entered.
iv. Asymptotic probability (steady-state probability) matrix ($\vec{\pi}$) can be calculated from $\vec{\pi}$=$(\pi_1, \pi_2, ..., \pi_n)^T$=$\begin{cases} \vec{\pi}P = \vec{\pi} \\ \sum_i \pi_i = 1 \end{cases}$.
v. Confidence interval (CI) for each transition CI=$Z_{\alpha/2}\sqrt{\frac{p_{i,j}(1-p_{i,j})}{n_i}}$ [35], where $p_{i,j}$ is the conditional probability of the transition, $Z_{\alpha/2}$ is from either the Normal or t-distribution, α is the significance level, and n_i is the times of state v_i.

1) State Merging Algorithm

In Algorithm 3, the pairwise Pearson χ^2 test is used for state merging. The test result shows whether two states are coming from the same state. The pair of most likelihood at one time is merged, and the merging is updated in the output count matrix. Then, the pairwise test is applied until all pairs reject the null hypothesis of two states from the same state. With the input of state transition count matrix M, state output matrix O, and significant confidence level α, the state merging is done as in Algorithm 3.

2) Model confidence test

After deriving a model from the data, it is needed to whether the data is enough to derive this model. If not enough, how much more data is needed? Thus, the model confidence test is applied (Algorithm 4 from [36]) to check the model.

With the input of transition probability matrix P, transition count matrix C, and asymptotic probability matrix $\vec{\pi}$, the test can be conducted as following Algorithm 4.

Algorithm 2 Inferring an HMM from sequence A with significance level α

```
1: for $n$ in $2:N$ do
2:     Infer $G_n$=$\langle A,V,E,P\rangle$ from the sequence $A$ with state-space parameter $L$=$n$.
3:     Merge states in $V$ using Algorithm 3 Pearson $\chi^2$ test.
4:     Do model confidence test for $G_n$. If does not enough, get more data and start over. Details of Model confidence test are described in Algorithm 4.
5:     Infer $G_{n+1}$=$\langle A,V',E',P'\rangle$ from the sequence $A$ with state-space parameter $L$=$n+1$.
6:     Merge states in $V'$ using Pearson $\chi^2$ test in Algorithm 3.
7:     Do model confidence test for $G_{n+1}$: If the training data is not enough, get more data and start over.
8:     Generate a long sequence $B$ from $G_{n+1}$ whose length longer than the result from model confidence test and from the generation method in Algorithm 5.
9:     Put the sequence $B$ into $G_n$. Get match probability matrix $F$.
10:    Get the Confidence Interval matrix $CI$ of $P$. Calculate $|P-F|-CI$, the elements less than zero in the result denotes the rejection proportion. Determine the rejection proportion by $p_{ij}=\sum_i d_{i,j}p_i$, where $d_{i,j}=D_{i,j}-CI$, $D=\{D\in(P-F)|D>0\}$, $p_i$ is the probability of state $i$ is entered.
11:    if $p_{ij}>\alpha$ then $n=n+1$ and go back to step 2
12:    else Set $G_n$ as the correct HMM for sequence $A$
13:    end if
14: end for
```

3) Generate a sequence of length l from an HMM G

In this section, the criteria for determining the proper value of L is explained, which terminates our inference process with the state-space that adequately models the underlying process. See [21] for a detailed description of the approach and associated proofs. The discussion is restricted to ergodic Markov processes, where all states are in a single strongly connected component. In Algorithm 5, a sequence of length l is generated from an HMM G for later convergence test.

4) Put sequence B into an HMM G

To test the convergence of G_n, the sequence B generated from the HMM G_{n+1} is put into the HMM G. For every state $v_i\in V$ of G, the state transition probability F is calculated in sequence B. If there is no transition in G for a window in sequence B to the next window, then record it as a rejection and go to the next window [21]. When sequences generated from a model created with state-space parameter L=$n+1$ are accepted by models generated from models generated using L=n, it is established that no extra information is extracted from the system. The state-space captured using state-space parameter value L=n adequately expresses the underlying process. Note that traditional HMM approaches start with an *a priori* known state-space and tune transition probabilities to make the data match the model. The approach described in this study learns the state-space and probabilities directly from the observed data. Algorithms 3 and 5 do not have an equivalence

Algorithm 3 State Merging

```
1: for each $i\in N$ do
2:     Test Pearson pairwise $\chi^2$ of independence [34] of rows in transition count matrix $M$.
3:     Denote the population proportion (or probability) falling in row $i$, column $j$ as $\pi_{ij}$. The total proportion for row $i$ is $\pi_i$. The total proportion for column $j$ is $\pi_j$. If the row and column proportions are independent, then $\pi_{ij}=\pi_{i.}\pi_{.j}$.
4:     The estimated expected value in row $i$, column $j$ is $E_{ij}=n\pi_{ij}$=$n\frac{n_{i.}}{n}\frac{n_{.j}}{n}$=$\frac{n_{i.}n_{.j}}{n}$.
5:     Test statistic is calculated as $\chi^2=\sum_{i,j}\frac{(n_{ij}-E_{ij})^2}{E_{ij}}$.
6:     Determine the $\chi^2_{\alpha,df}$ statistic for the $\chi^2$ test with significant level $\alpha$ and $df$=$(r-1)(c-1)$ where $r$=number of rows, $c$=number of columns.
7:     If $\chi^2\leq\chi^2_{\alpha,df}$ for any pairwise tests, the test accepts with significant level $\alpha$ the hypothesis that the two rows are from same state. Find the minimum value $\chi^2$ as $\chi^2_{min}$ and index $i$, $j$ $(i<j)$ of the pair of states it comes from. In the state transition count matrix $M$, add column $j$ to column $i$, add row $j$ to row $i$. Set zero of column $j$ and row $j$. In the state output count matrix $O$, add row $j$ to row $i$. Set zero of column $j$.
8:     Repeat steps 1-7 until $\chi^2$>$\chi^2_{\alpha,df}$ for all pairwise tests.
9:     Remove zero columns and zero rows in $M$ and $O$. Then quit with merged states transition count matrix and output count matrix.
10: end for
```

Algorithm 4 Confidence Test

```
1: $H_o$: data is not enough for any transitions, $H_a$: data is enough for any transitions.
2: Test statistic $Z$=argmin$_{p_{i,j}}(\frac{p_{i,j}}{\sqrt{\frac{p_{i,j}(1-p_{i,j})}{n_i}}})$ where 0<$p_{i,j}$<1 is the conditional probability of the transition, $n_i$=$\sum_j c_{i,j}$ is the total count of state $i$, $c_{i,j}$ is the element from transition count matrix $C$.
3: Rejection region: Reject $H_o$ if $z>z_\alpha$ that there is no need to collect more data. Otherwise, there is a need to collect more data. Enough data decision is calculated from sample size $D$=argmax$_{p_{i,j}}(\frac{z^2_\alpha(1-p_{i,j})}{p_{i,j}\pi_s})$ , where 0<$p_{i,j}$<1.
```

Algorithm 5 Generate a sequence of length l from an HMM G

```
1: Choose an initial state $v_o=v_i$ from state set $V$ where $v_i\in V$ and for $\forall v_i,v_j\in V,p(v_o=v_i)=p(v_o=v_j)$.
2: Using the probabilities of the outgoing transitions, select a transition $p_{i,j}$ to move to state $v_j$ from state $v_i$.
3: Record the label $a_i=E(v_i,v_j)$, where $a_i$ is associated with the chosen transition $p_{i,j}$.
4: Repeat steps 2 and 3 until $l$ labels have been recorded.
```

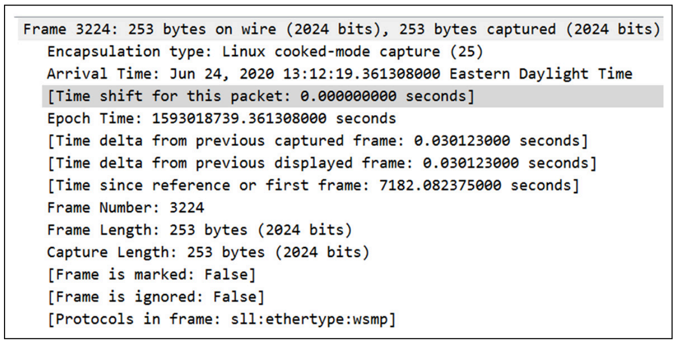
in the traditional approach. It is needed that these algorithms to determine when the state-space adequately expresses the underlying data.

III. Numerical Experiments

A. Roadside Unit Simulation

The simulations are run on DSRC devices as Fig. 3. On the first DSRC device, OBU processes are run. On the second DSRC device, the RSU process is run.

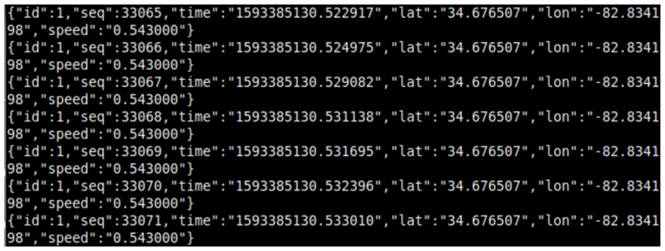
First, the simulation function is tested. In continuous four hours simulation, the RSU works well to avoid crashes. The traffic using *tcpdump* is captured. According to Fig. 5 of WSMP packet detail, the time shift for this packet approximately equals 0 s, which shows the real-time data exchange.



```
Frame 3224: 253 bytes on wire (2024 bits), 253 bytes captured (2024 bits)
    Encapsulation type: Linux cooked-mode capture (25)
    Arrival Time: Jun 24, 2020 13:12:19.361308000 Eastern Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1593018739.361308000 seconds
    [Time delta from previous captured frame: 0.030123000 seconds]
    [Time delta from previous displayed frame: 0.030123000 seconds]
    [Time since reference or first frame: 7182.082375000 seconds]
    Frame Number: 3224
    Frame Length: 253 bytes (2024 bits)
    Capture Length: 253 bytes (2024 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: sll:ethertype:wsmp]
```

Fig. 5. Example WSMP packet

In order to test the denial of service flaw of the DSRC, a flooding attack is performed at the DSRC channel wave-raw. The simulation is kept running with unexpected GPS information broadcasting on OBU at a very high-speed rate. Thus, RSU drops legitimate packets since the listening bandwidth is full with our flooding packets (Fig. 6). While the flooding attack is ongoing, several crashes are immediately detected (Fig. 7).

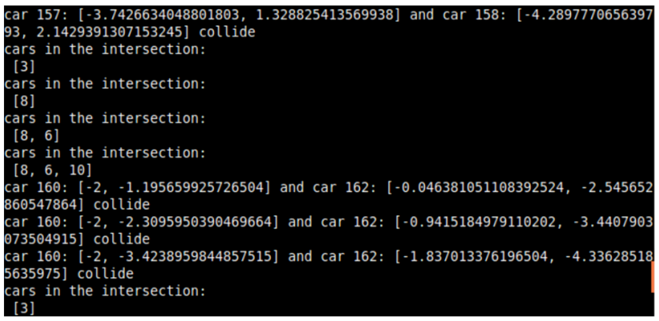


```
{"id":1,"seq":33065,"time":"1593385130.522917","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
{"id":1,"seq":33066,"time":"1593385130.524975","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
{"id":1,"seq":33067,"time":"1593385130.529082","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
{"id":1,"seq":33068,"time":"1593385130.531138","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
{"id":1,"seq":33069,"time":"1593385130.531695","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
{"id":1,"seq":33070,"time":"1593385130.532396","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
{"id":1,"seq":33071,"time":"1593385130.533010","lat":"34.676507","lon":"-82.8341
98","speed":"0.543000"}
```

Fig. 6. Example of flooding GPS information

B. Hidden Markov Models

A data set of 6433 packets is collected to build HMM. With the method described in Algorithms, the Timing HMM and Size HMM are obtained. From the HMM, the arrival of critical packets is aimed to be predicted, which are the instruction packets sent from RSU so that the attack can be conducted aiming at critical packets.



```
car 157: [-3.7426634048801803, 1.328825413569938] and car 158: [-4.2897770656397
93, 2.1429391307153245] collide
cars in the intersection:
 [3]
cars in the intersection:
 [8]
cars in the intersection:
 [8, 6]
cars in the intersection:
 [8, 6, 10]
car 160: [-2, -1.195659925726504] and car 162: [-0.046381051108392524, -2.545652
860547864] collide
car 160: [-2, -2.3095950390469664] and car 162: [-0.9415184979110202, -3.4407903
073504915] collide
car 160: [-2, -3.4238959844857515] and car 162: [-1.837013376196504, -4.33628518
5635975] collide
cars in the intersection:
 [3]
```

Fig. 7. Collision detected while doing flooding GPS information

1) Timing HMM

Timing symbolization is described in Table II, which shows that there we can distinguish between three timing ranges. This gives us a set of three symbols $\{a, b, c\}$ to represent the interpacket delays we observe. Initially, each symbol is an HMM state. By applying Algorithms 2, 3, and 4 we discover that the timing state-space converges when state-space parameter $L = 2$, as shown in Fig. 8. Parameter $L = 2$ means that the packet timing data depends on the last two packet types. This gives our HMM 9 states, $|\{a, b, c\}|^L$. Note that since Algorithm 3 can merge states, $|\{symbols\}|^L$ is the maximum size of the HMM state space.

Table IV includes state transition matrix. From the clear-text detail of the instruction packet (Fig. 8), it is recognized as a type a packet. So the prediction of a packet is considered in Timing HMM. According to Table IV, the packet leaving state S1 has the highest likelihood (0.8800) to be an a packet. And, the packet leaving state S5 has the second-highest likelihood (0.7300) to be an a packet.

TABLE IV. Timing HMM

	Transition Matrix		
State	a	b	c
S1	0.8708	0.0387	0.0905
S2	0.0204	0.2937	0.6859
S3	0.5774	0.2838	0.1389
S4	0.0383	0.6304	0.3313
S5	0.7318	0.0136	0.2545
S6	0.5723	0.2252	0.2025
S7	0.3922	0.2255	0.3824
S8	0.0460	0.6419	0.3120
S9	0.1732	0.5490	0.2778

2) Size HMM

Packet size symbolization is described in Table III. Packet size analysis resulted in only two symbols $\{a, b\}$ and as with the timing HMM resulted in $L = 2$. Since $|\{a, b\}|^L$, we get a state space of 4 as shown in Fig. 9. Table V includes state transition matrix. From the clear-text detail of the instruction packet (Fig. 9), it is recognized as a type y packet. So the prediction of y packet is considered in Timing HMM. According to Table V, the packet leaving state S3 has the highest likelihood (0.7101) to be a y packet.

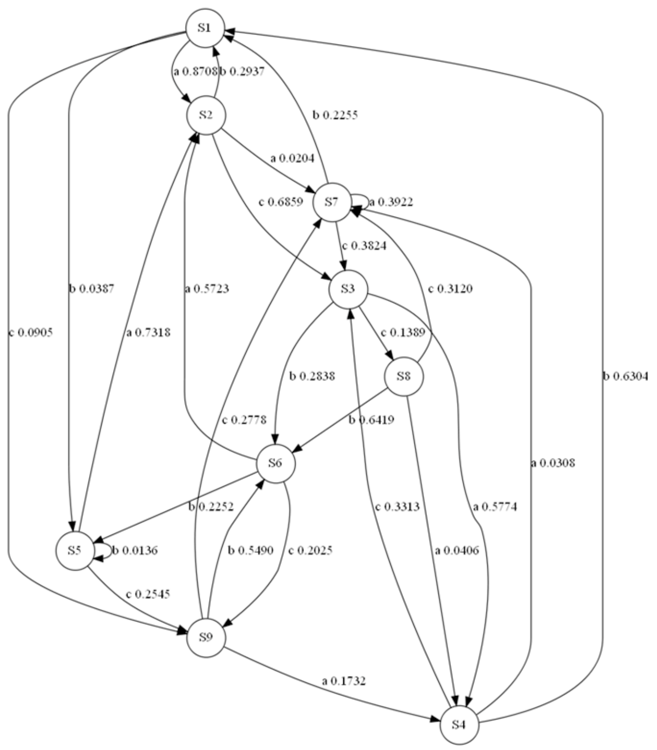


Fig. 8. HMM on timing

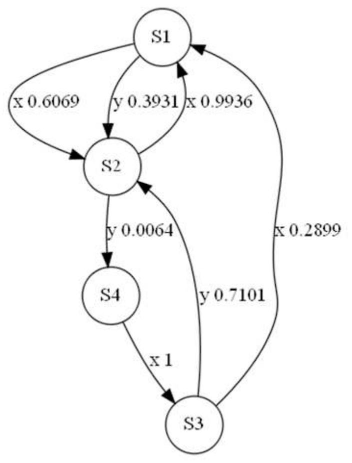


Fig. 9. HMM on packet-size

TABLE V. Packet-size HMM

	Transition Matrix	
State	x	y
S1	0.6069	0.3931
S2	0.9936	0.0064
S3	0.2899	0.7101
S4	1.0000	0.0000

C. Attack Simulation with HMM predictions

As shown in Table VI, there are three target states: timing state S1, timing state S5, and size state S3. An attack simulation is set up to test the HMMs prediction by dropping the packets leaving the target states.

The wave-raw process is used to simulate the DSRC/WAVE communication channel. The wave-raw process sends packets from the speed adjustment module and RSU to each other. For each received packet, the wave-raw process marks it with symbols as described in Table II and Table III. Since the HMMs obtained are both with space-state parameter L=2, a state could be identified after two received packets marked with symbols. With the identified state, the probability of the next packet type according to the HMMs can be predicted, and one can decide whether to drop the next packet. Six scenarios of experiments are set up. In each scenario, the wave-raw process would drop packets after different states.

TABLE VI. Target states

Target state	Category	States
1	Timing	S1
2	Timing	S5
3	Size	S3

The attacks are simulated under six different scenarios. The first scenario is a control group to see the crash rate if all packets from RSU dropped. In this scenario, the wave-raw process does not forward any packets from RSU to OBU. In the second scenario, the packet is dropped after the first timing state S1 is observed. In the third scenario, the packet is dropped after either timing state S1 or S5 is seen. In the fourth scenario, the packet is dropped after the size state S3 occurs. In the fifth scenario, the packet is dropped after any defined target states in Table VI. In the sixth scenario, the packet is dropped after the state is recognized as a combination of a target timing state and a size state. In these scenarios, a crash may occur when a car drives into the intersection while it should stop according to the information from RSU, but the information is dropped by an attack.

D. Datasets Obtained

After six scenarios (SCN) attack simulation experiments, the data is obtained as shown in Table VII. For each scenario, a total of 2000 cars approach the intersection. The crash is considered in an intersection with normal traffic flow and only between vehicles from different directions. Thus, vehicles are set that they come from four directions with Uniform interarrival times from $\mathcal{U}(2.1, 4.0)$ in seconds. The second column is the description of each scenario. The third column is the total packet number sent through DSRC/WAVE. The fourth column is the number of dropped packets. The fifth column shows the number of instructions in dropped packets. The sixth column is the number of crashes caused during each scenario.

E. Analysis of Results

Packets from RSU are marked as positive packets, packets from OBU as negative packets. The true positive (TP), true negative (TN), false positive (FP), and false negative (FN) of our attack as defined as:

- TP is the attack that drops a packet sent from RSU.
- TN is the attack does not drop a packet sent from OBU.
- FP is the attack that drops a packet sent from OBU.

- FN is the attack does not drop a packet sent from RSU.

False positive rate (FPR) is used to evaluate the reliability of the attack, where $FPR{=}\frac{FP}{FP+TN}$; True positive rate (TPR) is used to evaluate the sensitivity of the attack, where $TPR{=}\frac{TP}{TP+FN}$; The effect of the attack is presented by the crash proportion $p_c{=}\frac{crash\#}{drop\#}$. The rates are shown in Table VIII.

TABLE VII. Attack simulation

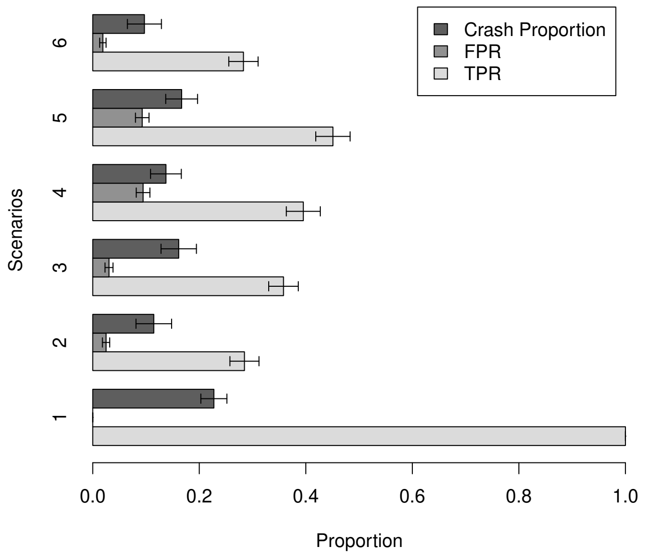
SCN	Attack target state	total packets	drop#	stop packet#	drop instruction#	crash#
1	Control group	3130	1130	1130	1130	257
2	most likely state	3051	349	1050	299	40
3	any target timing state	3147	471	1145	410	76
4	target size state	2904	546	903	357	75
5	any defined target state	2918	599	916	413	100
6	combination of target timing and size state	3032	330	1032	292	32

TABLE VIII. Analysis and comparison

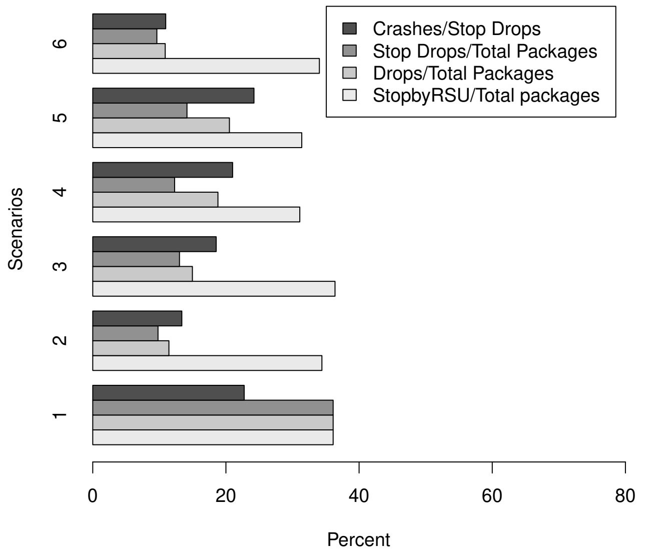
	SCN	Attack target state	Real CR%	Est. CR %	FPR%	TPR%	TP∩CR%
	1	Control group	22.74		0	100	
HMM	2	Max. likelihood timing state	11.46	19.48	2.50	28.48	6
HMM	3	Any target timing state	16.14	19.79	3.05	35.81	8
HMM	4	Target size state	13.74	14.87	9.45	39.53	9
HMM	5	Any defined target state	16.69	15.68	9.29	45.09	10
HMM	6	Combination of a target timing state and a size state	9.70	20.12	1.9	28.29	6
Random	Rate						
Random	0.1			7.96	10.38	9.67	2
Random	0.2			8.33	19.56	19.55	4
Random	0.3			8.67	29.28	31.17	7
Random	0.4			8.40	39.73	40.25	9
Random	0.5			8.31	49.94	49.75	11
Random	0.6			8.11	59.95	57.46	13
Random	0.7			8.24	70.48	69.17	16
Random	0.8			8.40	79.51	80.49	18
Random	0.9			8.30	90.43	89.86	20
SVM				16.49	20.27	72.50	16

To evaluate different attack scenarios, a column chart of true positive rate (TPR), false positive rate (FPR), and crash rate (CR) with the confidence interval (CI) for each scenario is plotted. The confidence intervals are calculated by $p \pm Z_{1-\alpha}\sqrt{\frac{p(1-p)}{N}}$, where $Z_{0.95} = 1.96$. The results are shown in Fig. 10a. Relative packet drop rates with respect to the total number of packages and stop packages are also shown in Fig. 10b for all scenarios. Approximately, stop package ratios sent by RSU are 40%. Crash rates with respect to stop packages are minimum at 10% in scenario 6 but reach up to 25% in scenario 5.

As shown in Fig. 10a, the control group is the first group of bars where the TPR is 100%, FPR is 0%, and crash proportion is 22.74%. This means that if one dropped only RSU packets and no other packets, this would cause vehicles to crash about 22.74% of the time. The goal of side-channel analysis is to cause vehicle crashes with fewer unnecessary packets dropping. Each scenario causes crashes and does a targeting attack. The FPRs are less than 10% for all scenarios. The effectiveness of side-channel analysis is proved. The attack scenarios are evaluated based on crash proportion and FPR.



(a) Scenario evaluation



(b) Packet Drop rates and crashes

Fig. 10. Analysis of results

First, the attacks based on one type of side-channel information are compared: timing side-channel attack for the most likely state, timing side-channel attack for two most likely states, and size side-channel attack. As shown in Fig. 10a, the 2^{nd} and 3^{rd} scenarios have the lowest FPR in 2^{nd}, 3^{rd}, and 4^{th} scenarios. With the windows of the confidence interval, there is no significant difference of FPR between the 2^{nd} scenario and the 3^{rd} scenario. Moreover, the 3^{rd} scenario also has the highest crash proportion. So one can conclude the 3^{rd} scenario of timing side-channel attack for two most likely states is best in side-channel analysis based on one type of information.

Then, all attack scenarios are compared to find the best attack method for this application. As shown in Fig. 10a, the 3^{rd} and 5^{th} scenarios have the highest crash proportion value, while the third scenario has a much lower FPR than the fifth scenario. So the 3^{rd} scenario is the best in five attack scenarios. In conclusion, timing side-channel analysis has better performance on predicted states. The attack targeting the packet leaving two most likely timing states worked best to cause crashes while avoiding the drop of unnecessary packets.

Note that random packet drops at different rates are also added. In Table VIII, crash percentages are presented. With random drops, it can be seen that only half of the crash percentages that of HMM can be achieved.

F. Comparison

With the SVM, from the 6433 collected packets, which is also the data to build HMM, two-thirds of them were used for training the classifier (4310 packets), and the remaining one-third was used as a test set. To validate the accuracy of the training set, k-fold cross-validation was used and showed that the base classification accuracy was 76.24% with a standard deviation of $\pm 2.12\%$. The classifier was then used on the remaining test data and resulted in an overall classification accuracy of 77.10%, a TPR of 72.50% for correctly identifying critical packets, and an FPR of 20.26%. This resulting test gives a much higher TPR rate for dropping packets from the RSU as opposed to the maximum TPR rate from scenario 5 in Table VIII (72.50% versus 45.09%). However, it also has a higher FPR rate (20.26% versus 9.29%)

For the three different predictor approaches: HMM, random, and SVM, a packet capture trace (pcap) of DSRC communications is utilized for training. As a consequence, both the TPR and FPR rates are directly related to empirical observations. For the crash rates, the HMM rates were empirically observed, but the random and SVM approaches relied on the assumption that the TPR probability is independent of the crash classification rate. That is $P(TP \cap crash) = P(TP)P(crash) = TPR \times CR$ where CR is the control's real crash rate from Table VIII. This assumption was used due to the loss of access to the DSRC testbed. Using this assumption allows us to estimate a baseline crash rate, but future work would need to validate this approach empirically. Note that randomly dropping 90% of all packets resulted in the highest TPR. However, the random predictor with high drop rates is much more trivial to detect, as it disrupts the majority of the packets sent. The SVM classifier results in a high TPR but still also has an FPR of 20%. It performs better in terms of FPR than the random drop (for large drop rates) but still results in dropping additional unnecessary packets. The HMM approach has a lower FPR rate and TPR, making it more stealthy and less sensitive than the SVM approach.

Note that the HMM approach reverse engineers the system that can observe given the fact that the packets are encrypted. It allows one to find the critical packets. SVM assumes that clear-text data is available for training. After training, the SVM classifies based solely on side-channels. Side-channel levels or features are used for monitoring the running system and making decisions at runtime which are used together with the clear-text packet captures to train SVM. At runtime, SVM uses only the side-channels. HMM uses only the side-channels for training. After training the states, one wants attacks to be found via clear-text or simply observing one instance of the process. At runtime, HMM uses only the side-channels. Thus, having access to the packet clear-text may give SVM has its advantage in getting a better TPR. As expected, random packet dropping performs the worst.

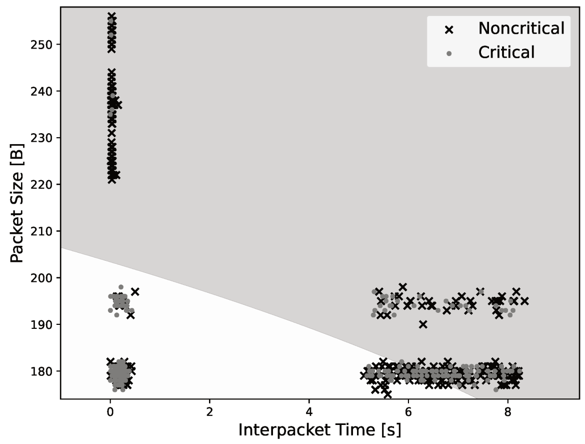


Fig. 11. The SVM Boundary classifies critical packets within the white region as critical (stop messages) and packets within the gray region non-critical (not containing stop or slow down messages). The true class of the point is represented by the point's color.

IV. Conclusions

This paper focuses on the evaluation analysis of DSRC/WAVE applications. To do this, a DSRC stop light application is set up based on a developed WSMP implementation. The data is sniffed through WSMP. The sniffed clear-text WSMP data content shows that the current implementation is insecure. Lack of security services, such as content encryption, makes it easy for attackers knowing critical car/road information with DSRC equipped devices. Then one can perform a DoS attack, successfully drop packets at the communication channel, and cause crashes.

Assuming all the security services would be implemented in the future, the "black box" attack is completed. Hidden Markov models (HMM) are constructed using sniffed inter-packet timing and packet size side channels since operational packets would be encrypted. An attack simulation is set up to test the HMM predictions of important packet arrival. The simulation results show the effect of the side-channel analysis. Timing side-channel analysis works better in the attack experiments.

The DoS result of packet dropping shows neither the application nor WSMP has a detection or prevention mechanism for DoS attacks. In DSRC communication, entropy-based DoS detection could be a good tool against DoS attacks. In DoS attack detection, entropy measures the amount of disorder in the observed data. For example, the roadside unit (RSU) system could calculate the entropy value of packet rate and packet size in this application. The RSU can also detect abnormal network traffic from the vehicle by cooperating with other RSU nearby. The vehicle volume could be estimated according to the information from other RSU. To prevent a DoS attack, DSRC should add the authentication mechanism to the standard.

To prevent side-channel attacks, the WSMP of DSRC should improve the packet formatting. For example, it could define the length of a packet through WSMP to prevent packet size side-channel attack.

In sum, DSRC is inherently vulnerable to DoS attacks since the wireless can be jammed or packet dropping attacks like the ones implemented in this study. SVM and HMM combination powerful for network protocol analysis to find DoS vulnerabilities which can be used for penetration testing of safety-critical protocols.

For future work, this study can be extended by

1) Validating the estimated crash rates for both the random and SVM predictors
2) Collecting more data and conduct the joint side channels analysis;
3) Applying this evaluation approach on more DSRC applications;
4) Testing other attack methods, e.g., radio signal jamming.

References

[1] X. Ma, J. Zhang, X. Yin, and K. S. Trivedi, "Design and analysis of a robust broadcast scheme for vanet safety-related services," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 46–61, 2011.

[2] R. C. Daniels, E. R. Yeh, and R. W. Heath, "Forward collision vehicular radar with ieee 802.11: Feasibility demonstration through measurements," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1404–1416, 2017.

[3] ARC-IT, "The national its reference architecture," $http://local.iteris.com/arc-it$, 2020, accessed: 2020-11-28.

[4] P. Fazio, F. De Rango, and C. Sottile, "A predictive cross-layered interference management in a multichannel mac with reactive routing in vanet," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1850–1862, 2015.

[5] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (v2x) communication platforms," *IEEE Transactions on Intelligent Vehicles*, 2020.

[6] J. Huang, D. Fang, Y. Qian, and R. Q. Hu, "Recent advances and challenges in security and privacy for v2x communications," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 244–266, 2020.

[7] İ. Özçelik and R. Brooks, *Distributed denial of service attacks: Real-world detection and mitigation*. CRC Press, 2020.

[8] D. Yang, K. Jiang, D. Zhao, C. Yu, Z. Cao, S. Xie, Z. Xiao, X. Jiao, S. Wang, and K. Zhang, "Intelligent and connected vehicles: Current status and future perspectives," *Science China Technological Sciences*, vol. 61, no. 10, pp. 1446–1471, 2018.

[9] C. Bettisworth, M. Burt, A. Chachich, R. Harrington, J. Hassol, A. Kim, K. Lamoureux, D. LaFrance-Linden, C. Maloney, D. Perlman *et al.*, "Status of the dedicated short-range communications technology and applications: report to congress," United States. Department of Transportation. Intelligent Transportation, Tech. Rep., 2015.

[10] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[11] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet computing*, vol. 10, no. 1, pp. 82–89, 2006.

[12] P. Kalra, K. Pandey, and A. Varshney, "Comparative analysis of syn flooding attacks on tcp connections," *Int. J. Inf. Comput. Technol*, vol. 4, no. 3, pp. 279–284, 2014.

[13] X. Zhong, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks, "Denial of service attack on tie-line bias control in a power system with pv plant," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 375–390, 2017.

[14] J. J. Blum and A. Eskandarian, "A reliable link-layer protocol for robust and scalable intervehicle communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 4–13, 2007.

[15] J. Thai, C. Yuan, and A. M. Bayen, "Resiliency of mobility-as-a-service systems to denial-of-service attacks," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 370–382, 2016.

[16] R. Merco, F. Ferrante, and P. Pisu, "A hybrid controller for dos-resilient string-stable vehicle platoons," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[17] Y. Ma, Z. Nie, S. Hu, Z. Li, R. Malekian, and M. Sotelo, "Fault detection filter and controller co-design for unmanned surface vehicles under dos attacks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[18] C. Laurendeau and M. Barbeau, "Threats to security in dsrc/wave," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2006, pp. 266–279.

[19] M. Islam, M. Chowdhury, H. Li, and H. Hu, "Cybersecurity attacks in vehicle-to-infrastructure applications and their prevention," *Transportation research record*, vol. 2672, no. 19, pp. 66–78, 2018.

[20] C. R. Shalizi, K. L. Shalizi, and J. P. Crutchfield, "An algorithm for pattern discovery in time series," *arXiv preprint cs/0210025*, 2002.

[21] J. M. Schwier, R. R. Brooks, C. Griffin, and S. Bukkapatnam, "Zero knowledge hidden markov model inference," *Pattern Recognition Letters*, vol. 30, no. 14, pp. 1273–1280, 2009.

[22] H. Bhanu, J. Schwier, R. Craven, I. Ozcelik, C. Griffin, and R. R. Brooks, "Noise tolerant symbolic learning of markov models of tunneled protocols," in *2011 7th International Wireless Communications and Mobile Computing Conference*. IEEE, 2011, pp. 1310–1314.

[23] B. Harakrishnan, S. Jason, C. Ryan, B. Richard R, H. Kathryn, G. Daniele, and G. Christopher, "Side-channel analysis for detecting protocol tunneling," *Advances in Internet of Things*, vol. 2011, 2011.

[24] X. Zhong, P. Arunagirinathan, A. Ahmadi, R. Brooks, and G. K. Venayagamoorthy, "Side-channels in electric power synchrophasor network data traffic," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, 2015, pp. 1–8.

[25] W. Yao, A. Yahya, F. Khan, Z. Tan, A. U. Rehman, J. M. Chuma, M. A. Jan, and M. Babar, "A secured and efficient communication scheme for decentralized cognitive radio-based internet of vehicles," *IEEE Access*, vol. 7, pp. 160 889–160 900, 2019.

[26] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[27] X. Liu, X. Fang, Z. Qin, C. Ye, and M. Xie, "A short-term forecasting algorithm for network traffic based on chaos theory and svm," *Journal of network and systems management*, vol. 19, no. 4, pp. 427–447, 2011.

[28] A. Y. Nikravesh, S. A. Ajila, C.-H. Lung, and W. Ding, "Mobile network traffic prediction using mlp, mlpwd, and svm," in *2016 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2016, pp. 402–409.

[29] S. Dong, "Multi class svm algorithm with active learning for network traffic classification," *Expert Systems with Applications*, vol. 176, p. 114885, 2021.

[30] K. Kato and V. Klyuev, "An intelligent ddos attack detection system using packet analysis and support vector machine," *IJICR*, vol. 14, no. 5, p. 3, 2014.

[31] L. Yu, J. M. Schwier, R. M. Craven, R. R. Brooks, and C. Griffin, "Inferring statistically significant hidden markov models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1548–1558, 2012.

[32] C. Griffin, R. R. Brooks, and J. Schwier, "A hybrid statistical technique for modeling recurrent tracks in a compact set," *IEEE transactions on automatic control*, vol. 56, no. 8, pp. 1926–1931, 2011.

[33] P. Ryan, S. A. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe, *The modelling and analysis of security protocols: the csp approach*. Addison-Wesley Professional, 2001.

[34] R. Ott and M. Longnecker, "Multiple comparisons, an introduction to statistical methods and data analysis (pp. 438–440)," *Australia: Duxbury Thomson Learning*, 2001.

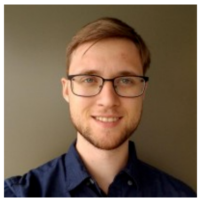
[35] J. M. Schwier, R. R. Brooks, and C. Griffin, "Methods to window data to differentiate between markov models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 3, pp. 650–663, 2010.

[36] C. Lu, J. M. Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin, "A normalized statistical metric space for hidden markov models," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 806–819, 2013.

Fei Sun Fei Sun received a B.S. degree in Electronic Information Science and Technology from Xiamen University, Fujian, China and a M.S. degree in Department of Electrical and Computer Engineering, Clemson University, Clemson, South Carolina. Her research interests include queuing theory, cyber-security, connected vehicle communication. She is currently with NSFOCUS Technologies Group Co., Ltd, working on network security services.

Richard R. Brooks is Professor of Electrical and Computer Engineering at Clemson University in Clemson, South Carolina, He received a PhD in Computer Science from Louisiana State University and a B.A. in Mathematical Sciences from The Johns Hopkins University. Dr. Brooks also studied Operations Research at the Conservatoire National des arts et Metiers in Paris, France. He is a senior member of the IEEE. He is fluent in German and French. Dr. Brooks' research on computer and network security has been sponsored by US DoD, NIST, Dept. of State, NSF, and BMW. His security research works to advance freedom of expression and protect vulnerable civilian populations.

Gurcan Comert received the B.Sc. and M.Sc. degree in Industrial Engineering from Fatih University, Istanbul, Turkey and the Ph.D. degree in Civil Engineering from University of South Carolina, Columbia, SC, in 2003, 2005, and 2008 respectively. He is currently with Physics and Engineering Department, Benedict College, Columbia, SC. His research interests include applications of statistical models to transportation problems, traffic parameter prediction, and stochastic models.



Nathan Tusing received the B.S. degree in engineering from Bob Jones University, Greenville, South Carolina in 2018. He is currently pursuing the Ph.D. Degree in computer engineering from Clemson University, Clemson, South Carolina with the Holcombe Department of Electrical and Computer Engineering. His research interests include network side channel analysis, statistical learning, and combinatorial game theory.