GPSKey: GPS-based Secret Key Establishment for Intra-Vehicle Environment

Edwin Yang
University of Oklahoma
Norman, OK, USA
edwiny@ou.edu

Song Fang
University of Oklahoma
Norman, OK, USA
songf@ou.edu

***Abstract*—With the advent of the in-vehicle infotainment (IVI) systems (e.g., Android Automotive) and other portable devices (e.g., smartphones) that may be brought into a vehicle, it becomes crucial to establish a secure channel between the vehicle and an in-vehicle device or between two in-vehicle devices. Traditional pairing schemes are tedious, as they require user interaction (e.g., manually typing in a passcode or bringing the two devices close to each other). Modern vehicles, together with smartphones and many emerging Internet-of-things (IoT) devices (e.g., dashcam) are often equipped with built-in Global Positioning System (GPS) receivers. In this paper, we propose a GPS-based Key establishment technique, called *GPSKey*, by leveraging the inherent randomness of vehicle movement. Specifically, vehicle movement changes with road ground conditions, traffic situations, and pedal operations. It thus may have rich randomness. Meanwhile, two in-vehicle GPS receivers can observe the same vehicle movement and exploit it for key establishment without requiring user interaction. We implement a prototype of *GPSKey* on top of off-the-shelf devices. Experimental results show that legitimate devices in the same vehicle require 1.18-minute of driving on average to establish a 128-bit key. Meanwhile, the attacker who follows or leads the victim's vehicle is unable to infer the key.**

I. Introduction

Global Positioning System (GPS) receivers are popularly available in modern vehicles and various devices (e.g., smartphones, tablets, cameras, or IoT devices). European Global Navigation Satellite Systems Agency expects the number of installed Global Navigation Satellite System (GNSS) receivers will exceed 8 billion by 2025 [2], where GNSS includes GPS (USA), Galileo (Europe), GLONASS (Russia), and BeiDou (China). As such devices often carry privacy-sensitive information, they are required to establish secure communication channels [7], [17], [33]. For example, there is a growing number of smart devices inside modern cars [6], and users may need to pair such a device with the vehicle's IVI system or another in-vehicle device. Existing device pairing schemes, however, are not user-friendly and may be unsafe to be performed while driving. For example, the existing IVI systems require a user to initiate the pairing process (e.g., activate discovery/search mode), select a target device, and type a PIN to verify the pairing procedure.

Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2022
24 April 2022, San Diego, CA, USA
ISBN 1-891562-75-4
https://dx.doi.org/10.14722/autosec.2022.23023
www.ndss-symposium.org

In order to overcome such limitations, context-based pairing schemes, where they extract secret key from common observations (e.g., visual channel [29], ambient audio [9], [14], timing of detected events [11], wireless signal strength [12], [33] and wireless channel interference [19], [22]) without requiring human interactions are proposed. However, these schemes present two shortcomings: (1) The surrounding environment may not consistently provide enough randomness for effective key generations. For example, [11] requires a user to generate additional events to reduce pairing time. (2) Some channels may not available in a normal vehicle. For example, audio signals in a vehicle can be easily interfered with by noises and vibrations.

Recent research efforts propose context-based pairing schemes for vehicular environments [6], [16]. They focus on a fact that devices located in the same vehicle experience highly similar movement, which can be captured by inertial measurement unit (IMU) sensors (e.g., accelerometer, gyroscope, and magnetometer). For example, [16] derives a key from the vertical acceleration (perpendicular to lateral and longitudinal acceleration) of a vehicle, which is affected by vehicle type and road condition. [6] employs three different sensors, where a vehicle's lateral and longitudinal accelerations are captured by an accelerometer, a gyroscope is utilized for measuring the vehicle's turns, and a barometer monitors altitude changes. However, these schemes present two limitations: (1) IMU sensor data can be easily disturbed by device movement, thus may not be robust or provide a consistent performance when the user uses or wear the device during key establishment. (2) IMU sensor data are considered insensitive on modern mobile devices, where any mobile applications can access them without specific privilege [21], [35].

In this paper, we utilize a vehicle's movement pattern in GPS data to derive secret keys. The key idea is that two GPS-enabled devices in the same vehicle may observe the same vehicle movement with highly similar accuracy. The vehicle movement is determined by a driver's pedal control, which contains randomness from the surrounding traffic environment. Each device extracts secret bits from the observed vehicle velocity when the vehicle is moving. While the devices may get slightly different binary bit sequences due to observation errors, they perform reconciliation to agree on the same key.

We implement a prototype of the proposed key establishment scheme and perform real-world experiments to quantify its performance. Extensive experimental results show that the legitimate devices can establish a 128-bit key in a 71-

second driving period on average, while the attacker who follows or leads the target vehicle fails to acquire similar GPS observations. We also note that the proposed scheme can be extended to other GNSSs (e.g., GLONASS, Galileo, BeiDou).

We summarize key contributions of this paper as follows:

- We propose a GPS-based key establishment scheme that utilizes permission-protected GPS data that contain randomness from driving as the secret key source.
- We design an algorithm that extracts randomness from the observed vehicle movement in GPS data.
- Our real-world experiment results show that the proposed scheme successfully establishes secret keys between legitimate GPS-enabled devices.
- We demonstrate that an attacker who follows or leads a target vehicle cannot recover the same secret keys as the legitimate devices.

The rest of the paper is organized as follows. We present assumptions and the adversary model in Section II. We present the detailed key establishment scheme in Section III, and demonstrate real-world experiment results in Section IV. Next, we summarize related work that is closely related to our scheme in Section V, and conclude the paper in Section VI.

II. Adversary Model

We consider a general scenario, where two legitimate GPSenabled devices within the same vehicle aim to establish a secure channel by deriving the same secret key. The two devices have no prior knowledge about each other, and the user is free to move or use the devices during driving.

In our adversary model, we consider an attacker who aims to recover the same key as the legitimate devices in the target vehicle. We assume that the adversary knows how the proposed secret key establishment works, however she cannot be within the target vehicle during the key establishment process. We note that this is a common assumption in other context-based secret key establishment schemes [11], [6], [16], where a moving vehicle provides a physical security boundary. Instead, the adversary attempts to observe the target vehicle's velocity from outside of the target vehicle to get the same key as the legitimate devices. Specifically, we consider the following two attack scenarios:

- **Stalking Attack:** The adversary follows the target vehicle to obtain similar velocity observation as the legitimate device. Instead of utilizing dedicated speed measurement systems, the attacker acquires velocity data using a GPS-enabled device (e.g., smartphone).
- **Leading Attack:** The adversary obtains velocity data while the target vehicle is following the adversary (e.g., follows the vehicle driven by the adversary). In this scenario, the attacker aims to acquire highly similar velocity observation as the legitimate devices by controlling or limiting the target vehicle's movement.

Meanwhile, GPS is vulnerable against spoofing attacks where the adversary tries to inject falsified GPS signals over the air to the victim's GPS-enabled devices [34], [31]. *GPSKey* can be targeted by such attacks, where the attacker tries to make the legitimate devices to establish a key that can be manipulated by the attacker herself. However, we note that a GPS receiver may cross-check the collected GPS observations with dead reckoning results based on IMU sensors (e.g., accelerometer) [34] or with the location acquired from the cellular network to detect GPS spoofing attacks [25].

III. Design of GPSKey

We present the architecture of the proposed key generation scheme by introducing the three key processes: *data pre-processing*, *randomness extraction*, and *key generation*.

A. System Overview

GPSKey aims to make two GPS-enabled devices located in the same vehicle establishing the same key. To achieve this, we focus on the GPS velocity measurement obtained by each of the devices. First, the legitimate devices co-located in the same vehicle continuously record GPS velocity data when the vehicle is moving. The variations in the observed velocity data are affected by the randomness that comes from the road ground conditions, the traffic situations. Next, each device removes noises in the collected velocity data to deal with GPS velocity observation error. Then, the devices agree on a starting point of the key extraction, and the driving data that do not present enough randomness are filtered. Each device then extracts a secret bitstream from the velocity data with enough randomness. As there may exist mismatching bits between Alice's and Bob's bitstreams, a reconciliation process is performed. Figure 1 demonstrates how *GPSKey* works.

B. Data Pre-processing

Due to environmental noise and GPS signal loss, velocity observations between the legitimate devices may introduce inconsistency. Thus, we first perform interpolation on each device to enable each device to achieve a consistent sampling rate. Next, each device performs noise reduction to obtain more accurate velocity observation.

1) Interpolation: Each GPS receiver obtains velocity measurement from the GPS signals sent by the GPS satellites. However, due to various environmental factors (e.g., weather, buildings), there is no guarantee that the devices always obtain GPS data samples at a constant rate. In real-world driving environments, we empirically observe that the GPS-enabled devices occasionally obtain null GPS data samples. To compensate for the empty samples, and considering that the velocity of a vehicle is not drastically changed, we perform linear interpolation to estimate the missing samples.

Meanwhile, typical GPS receivers usually present a low sampling rate (1 Hz). As this may slow down the key generation process, we perform up-sampling to the obtained GPS velocity data using linear interpolation. Thus, each device achieves a 2 Hz of GPS data sampling rate. We note that up-sampling is a widely utilized pre-processing method for secret key establishment schemes [28], [20]. Although we perform up-sampling to GPS velocity data, the correlation between the generated secret bits and the velocity data are removed after performing quantization (i.e., the GPS velocity data cannot be restored from the corresponding bit sequence).

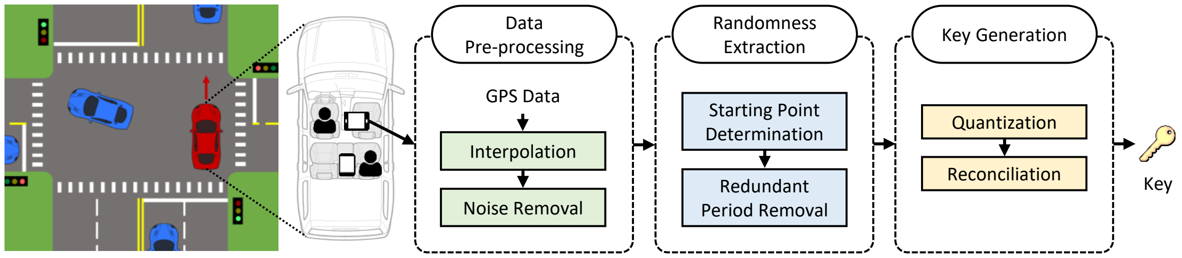

Fig. 1: System overview of *GPSKey*.

2) Noise Reduction: We observe that the vehicle movement-related variations in GPS data are primarily located at low frequencies. To preserve valuable velocity variations and mitigate random high-frequency noises introduced by environmental interference or hardware imperfection, we apply Butterworth low-pass filter [26].

C. Randomness Extraction

To agree on the same secret key, two legitimate devices need to synchronize the starting point of the secret bit generation. Meanwhile, when the vehicle is stopped or cruising for a long time, the corresponding velocity observation may not present enough randomness, and the resultant secret key may be easily inferred (e.g., a long sequence of 0 or 1). To mitigate such cases, the randomness extraction phase extracts velocity observations that present enough randomness by utilizing a stop and cruise period detection algorithm.

1) Starting Point Determination: In the real-world driving scenario, a vehicle may require some time to acquire GPS data after ignition (e.g., required time to acquire initial GPS data sample), and a vehicle moving at too low speed may not present enough randomness in the corresponding GPS data. To avoid collecting GPS data under such cases, two legitimate devices record potential starting timestamps when the vehicle velocity exceeds a threshold V_{thr} for the first time. However, there is no guarantee that the two devices always agree on the same starting time due to velocity observation errors. To ensure that the legitimate devices get the same starting point, we allow one device that initiates the key establishment process to notify the obtained starting point (i.e., GPS timestamp) to the other device. We note that GPS timestamps are accurate, as clocks GPS are based on high precision atomic clocks, and they are periodically corrected and synchronized by the Master Control Station on the earth [10].

2) Redundant Period Removal: When the vehicle is stopped or cruising for a long period, the corresponding velocity observations may not present enough randomness. To mitigate such cases, we propose stop and cruise period detection algorithms.

Stop Period Detection: When the vehicle is stopped, we observe that the obtained velocity data present non-zero (but close to zero) velocity or null values. For this reason, entirely relying on the velocity data may not be accurate. Thus, we utilize both location (i.e., GPS coordinates) and velocity to determine if the vehicle is stopped. Let v_t and d_t denote thresholds that denote maximum observation errors for velocity and coordinate, respectively, when the vehicle is stopped. By investigating if the observed velocity is lower than v_t and the coordinate change is smaller than d_t, we determine the stopped state of the vehicle.

Algorithm 1 Cruise Period Detection

```
Input: A velocity stream $\mathbf{v}$ with $N$ samples; pre-defined
    maximum velocity variance $v_{max}$, sliding window size $w$,
    target cruising duration $d$
Output: A set of cruising periods $C$
 1: $i \leftarrow 1$
 2: while $i < N$ do
 3:     $c \leftarrow [v_i, \cdots, v_{i+w-1}]$
 4:     $\bar{v} \leftarrow \frac{1}{w}\sum_{j=i}^{i+w-1} v_j$
 5:     $\sigma \leftarrow \frac{1}{w}\sum_{j=i}^{i+w-1}(v_j - \bar{v})^2$
 6:     if $w > d$ then                          # Cruise period detected
 7:         $C \leftarrow C \cup \{c\}$
 8:         $i \leftarrow i + w$
            continue
 9:     end if
10:     if $\sigma < v_{max}$ then
11:         $w \leftarrow w + 1$
            continue
12:     else
13:         $i \leftarrow i + 1$
14:     end if
15: end while
```

Cruise Period Detection: To identify if the vehicle is traveling at a constant velocity, we first define the maximum velocity variance v_{max} which denotes the maximum velocity fluctuations during a cruising period. We note that v_{max} can be obtained by profiling cruising velocity data.

We employ Algorithm 1 to identify cruise periods in a series of velocity observations consists of N samples, $\mathbf{v} = [v_1, v_2, \cdots, v_N]$. Algorithm 1 first checks if the candidate period c presents small variations by comparing its variance σ and the threshold v_{max}. If $\sigma < v_{max}$, it increases the window size w by 1 and compare σ and v_{max} again. If w exceeds a pre-defined cruise period d, c is then considered as a cruise period, thus is added to a cruise period set C.

After running both stop period and cruise period detection, each device obtains r redundant periods $R_v = [(t_{s1}, t_{e1}), \cdots, (t_{sr}, t_{er})]$ from the velocity data, where t_{si} and t_{ei} denote the start and end timestamps of i^{th} redundant period ($i \in \{1, \cdots, r\}$). From the observed velocity data stream $\mathbf{v}$, the redundant periods R_v are then removed. However, due to errors in GPS velocity measurement, there is no guarantee that the legitimate devices always identify the same redundant periods. To resolve this problem, we allow one party to share the identified redundant periods with the other party.

D. Key Extraction

In the key extraction phase, each device extracts secret bits from the processed GPS velocity data by performing quantization. As two legitimate devices may not achieve the same key after the quantization process, a secure sketch [5] is utilized to correct the bit mismatches.

1) Quantization: To extract secret bits from a series of velocity samples, we first investigate traditional value-based quantization techniques that utilize a single threshold. Without loss of generality, if a sample value exceeds a pre-determined threshold value, the sample is encoded as 1, otherwise, it is encoded as 0. However, applying a single threshold over a velocity data stream may return a series of long, consecutive 1 or 0 sequences, thus they cannot be utilized.

Instead of utilizing a single threshold over whole input data, we quantize the input data by applying a variable threshold. For a velocity stream $V = [v_1, \cdots, v_N]$, we employ a moving window of k-second period to determine the threshold T_Q. For a k-second period $V' = [v_i, \cdots, v_{i+k-1}]$, we define $T_Q = \frac{1}{k}\sum_{j=i}^{j+k-1} V'$, i.e., average velocity. For each sample v_i in V', we extract a bit as follows:

$$f(v_i) = \begin{cases} 1, & T_Q > v_i \\ 0, & \text{otherwise} \end{cases}, \quad (1)$$

After extracting k-bits, we move the window to the next k-samples (i.e., from $(i+k)^{th}$ to $(i+2k-1)^{th}$ samples), and iterate the process until all samples are quantized.

2) Reconciliation: Due to hardware imperfections in GPS receivers and environmental interference, both legitimate devices may experience a small number of mismatching bits after obtaining their initial bitstreams. To make Alice and Bob agree on the same key, we employ a secure sketch [5], which can correct the mismatching bits between the legitimate devices.

A secure sketch is a pair of functions, including the sketching function SS and the recover function REC. SS outputs a public sketch s from its input w, revealing little information about w. Meanwhile, REC takes s and another input w' to exactly obtain w. To correct the mismatch between w and w', the construction of a secure sketch is based on an error correction code (ECC). The idea is to use ECC to correct errors in w by shifting the code so that a codeword matches up with w and sharing the shift as the sketch s.

Let one legitimate device, Alice, obtains a bitstream w, while the other legitimate device, Bob gets w', where w and w' present a small number of mismatching bits. Then, Alice and Bob perform the following process:

(1) Alice generates a sketch $s = SS(w) = w - c$, where c is a random ECC codeword.
(2) Alice sends s to Bob via the public channel.
(3) Bob performs $c' = w' - s$, and decode c' to obtain c.
(4) Bob recovers $w = c + s$, i.e., $REC(w', s) = w$.

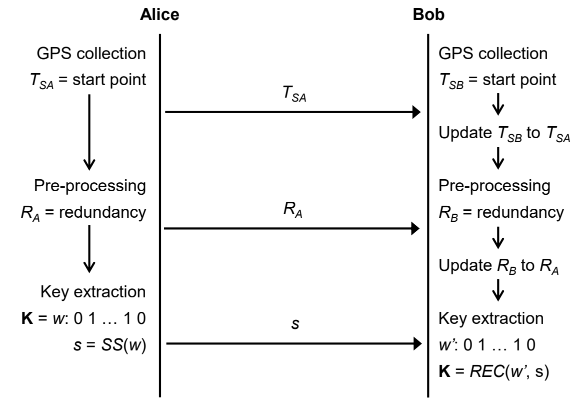

Fig. 2: Detailed flow diagram of *GPSKey* protocol.

Without loss of generality, we utilize the basic code offset construction [5] and employ Reed-Solomon (RS) code, i.e., $RS(n, k)$ which has n symbols of p bits each. The first k of the n symbols are information bits, and the rest $n - k$ bits are parity bits. The parity bits are derived based on the RS algorithm and the information bits. Given a symbol size l, the maximum length of the codeword C is $m = 2^p - 1$, thus $n \leq m$. Correspondingly, the RS decoder can correct any $\frac{n-k}{2}$ symbol errors in the codeword. Figure 2 presents the flow diagram of *GPSKey*.

E. Discussion

IMU-based Velocity Estimation: IMU sensors (e.g., accelerometer) can also be utilized to measure the vehicle velocity and popular mobile operating systems often give third-party applications unrestricted access to such sensors. One concern is thus whether an adversary can infer the secret key via the measurements of IMU sensors. To estimate accurate vehicle velocities using IMU sensors, the attacker must first pre-know the orientation of the target device. Also, the device cannot be moved during the key establishment process. Both requirements impose a great hurdle for the attacker. Moreover, velocities obtained through IMU sensors or GPS may not match due to unpredicted GPS observation errors.

Integration into V2X Communication Systems: In the IEEE 1609.2 Vehicle-to-everything (V2X) Standard, in order to improve the information that other nearby drivers have about the traffic conditions in their immediate vicinity, each on-board unit (OBU) device would broadcast a basic safety message (BSM) (including the vehicle's current GPS-based position and velocity) at a rate of 10Hz [1]. Such meta-information, however, may endanger the privacy of drivers, enabling a passive eavesdropper to trace a vehicle via a consistent series of BSMs belonging to the vehicle [4], [8]. One of the primary means for achieving privacy in V2X systems is to use group signatures that rely on pseudonyms (short-lived identifiers) [1], [3]. During the key establishment period, if multiple vehicles around the attacker have the same group identifier with the target vehicle, the attacker may not be able to link two different BSMs (transmitted at the two locations) to the target vehicle. As a result, without the full GPS velocity time series of the target vehicle for key generation, it would be difficult for the adversary to generate the same key with the legitimate devices.

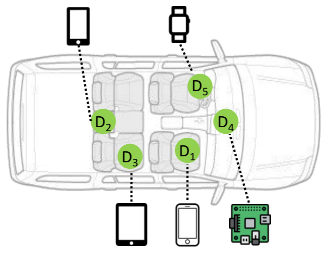

Fig. 3: Different devices placed at different locations.

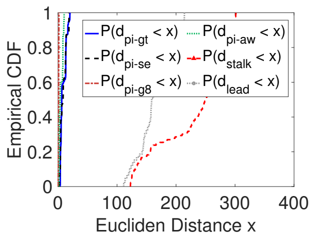

Fig. 4: The empirical CDFs of measurement similarities.

Meanwhile, if every vehicle (including the target) within a geographic area were in its own group, the group identifier thus becomes a unique vehicle identifier. In this extreme case, *GPSKey* should not be launched as the adversary would get access to the same GPS velocity times series with the target vehicle and then be able to infer the same key with the legitimate devices.

IV. Experimental Evaluation

A. Experiment Setup

We implement a prototype of *GPSKey* on commercial GPS-enabled devices as well as a popular GPS module in navigation systems (i.e., Ublox NEO-6M [32]). In our experiments, two legitimate devices are placed in a common vehicle. Each device could be an Android device (LG G8X ThinQ smartphone, Samsung Galaxy Tab S7 tablet), an iOS smartphone (iPhone SE), a smartwatch (Apple Watch 3) on a driver's wrist, or Raspberry Pi connected with a NEO-6 GPS module. We launch *GPSKey* when the vehicle is moving on urban and rural roads. Meanwhile, an attacker drives another vehicle with an adversary GPS-enabled device. The attacker follows or leads the target vehicle to obtain similar driving data as the legitimate devices, and attempts to recover the same secret key shared between the legitimate devices. In our attack experiments, we assume a stronger attacker, who can apply optimal time offset to the collected driving data to minimize mismatch between the attacker's driving data and the driving data collected by the legitimate devices.

Experimental Metrics: We quantify the performance of the proposed scheme by utilizing the following experimental metrics that are frequently utilized in the existing key generation schemes (e.g., [13], [17], [19]):

- *Secret Bit Rate (SBR):* SBR is the number of secret bits generated per second. Thus, SBR determines the required driving duration to establish a secret key with the required key length.
- *Bit Error Rate (BER):* BER is the percentage of mismatched bits between the initial keys after the quantization phase at the two devices. The lower BER reduces the burden on the reconciliation phase.
- *Bit Randomness:* To verify if generated bitstreams present enough randomness, we utilize the standard NIST randomness test suite [24].

B. Impact of Device Type and Proximity

GPSKey takes advantage of the fact that GPS receivers on legitimate devices present highly similar accuracy in the same vehicle. If this fact does not hold, generated bitstreams may present a high number of mismatching bits, where the devices would fail to acquire the same key. To verify this, we place the legitimate devices at different in-vehicle locations as shown in Figure 3. We place iPhone SE, LG G8X ThinQ, Samsung Galaxy Tab S7, and Raspberry Pi with NEO-6M at positions D_1, D_2, D_3 and D_4, respectively, and let the driver wear an Apple Watch 3 on the left wrist during driving (depicted as D_5). We let the user drive a compact vehicle under real-world roads while an attacker drives her vehicle to launch the stalking and leading attacks. For each driving, we collect 300 GPS samples and perform the experiment 100 times.

Figure 4 depicts the empirical cumulative distribution functions (CDFs) of Euclidean distances between legitimate device pairs as well as between a legitimate device and an attacker's device. For the legitimate devices, we denote Raspberry Pi, Samsung Galaxy Tab S7, iPhone SE, LG G8X ThinQ, and Apple Watch 3 as pi, se, $g8$, and aw, respectively. For the stalking and leading attack scenarios, we denote the Euclidean distance between the attacker's observation and the legitimate user's observation as d_{stalk} and d_{lead}, respectively.

We can see that 95.6% of the Euclidean distances between the legitimate devices are less than 18.5, while d_{stalk} and d_{lead} are at least 111.2 and 122.7, respectively, thus the attacker fails to observe similar driving data. The result shows that the GPS-enabled devices in the same vehicle observe highly similar velocity data. We note that a smartwatch on the driver's wrist also presents highly similar driving data, demonstrating the practicality and robustness of the proposed scheme.

C. Driving Data Uniqueness

While *GPSKey* derives secret keys from the observed vehicle movement on the road, the uniqueness of the generated keys depends on the uniqueness of the corresponding driving data. To verify the feasibility of *GPSKey*, we explore if collected GPS data for each trip present unique property. We allow a target user to drive on the same typical urban road 100 times while collecting driving data with two legitimate devices (Dev_A and Dev_B). We derive Euclidean distance between the GPS velocity data from different trips to verify the uniqueness of GPS velocity data.

Figure 5 demonstrates single-trip GPS trajectories on the map at the two legitimate devices. To preserve the privacy of the driver, we do not show the corresponding longitude and latitude values. We can see that the legitimate devices acquire highly similar GPS information during the trip. Figure 6 plots the empirical CDFs of the Euclidean distance d_{legit} between the velocity data of Dev_A and Dev_B, as well as d_{self} between the velocity data obtained by the same device for different trips on the same road. We observe that d_{self} is greater than 65.97 with a probability of 0.95, while d_{legit} is smaller than 14.73 with a probability of 0.95. The result shows that each trip on the same road would generate quite a unique velocity stream due to uncontrollable environmental factors (e.g., weather) and traffic situations (e.g., traffic lights).

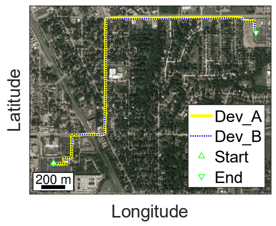

Fig. 5: GPS trajectories of two legitimate devices.

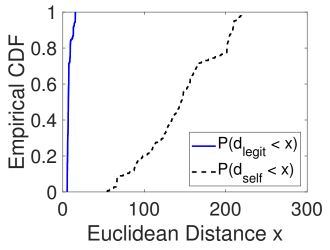

Fig. 6: The empirical CDFs of d_{legit} and d_{self}.

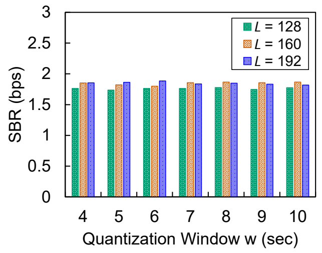

Fig. 7: Secret bit rate vs. w (legitimate devices).

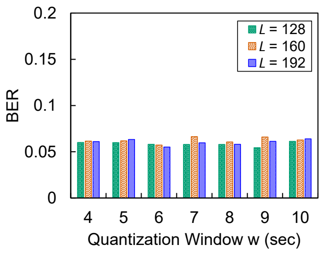

Fig. 8: Bit error rate vs. w (legitimate devices).

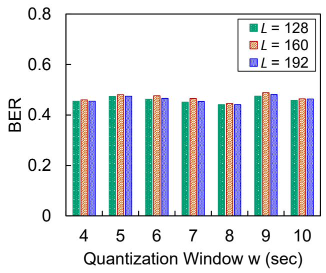

Fig. 9: Bit error rate vs. w (stalking attack).

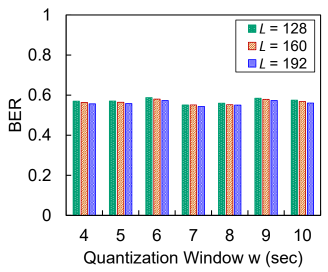

Fig. 10: Bit error rate vs. w (leading attack).

TABLE I: p-Value of NIST Test Result

NIST Test	p-value
Frequency	0.2159
Block Frequency	0.3397
DFT	0.1443
Longest Run	0.2986
Cumulative Sums	0.3146
Runs	0.0353
Serial	0.0461
Approximate Entropy	0.0546

D. Key Generation Performance

We evaluate the performance of *GPSKey* with different key generation parameters, as well as its resistance against the stalking and leading attack. To derive a secret key, the size w of a quantization window must be determined. We note that w is related to the randomness of the generated bit sequence. On one hand, too small w may introduce an increased bit error rate as the threshold T_Q may not accurately represent the average velocity of each w-second period. On the other hand, using too large w may result in a long sequence of 1 or 0, as using a large w is the same as applying a fixed threshold.

Considering the randomness of the generated bitstream, we investigate the impact of different w on key generation performance by varying $4 \leq w \leq 10$ in increments of 1. For each w, we perform 100 attempts of generating a key with length of L ($L \in \{128, 160, 192\}$). For the stalking and leading attacks, we also attempt 100 key establishments.

Legitimate Key Establishment: Figures 7 and 8 illustrate the average secret bit rate and bit error rate between the bitstreams generated between a pair of legitimate devices before the reconciliation phase is executed. We have the following observations. First, we can see that average SBR ranges from 1.74 to 1.89, where *GPSKey* presents consistent SBRs for different quantization window size w and key length L. We note that the achieved SBRs are smaller than the ideal SBR (i.e., 2.0 bps). This is due to inevitable redundancy in the driving data, where the driver needs to stop the vehicle or maintain constant velocity on cleared roads. Second, we observe that the average BER between legitimate devices ranges from 0.05 to 0.07, which indicates that the devices derive highly similar bitstreams from their GPS observations.

Stalking/Leading Attack Resistance: In Figure 9 and 10, we can see that the adversary achieves much higher BER with stalking attack (0.44 - 0.49) and leading attack (0.54 - 0.59), respectively. The results show that the attacker cannot achieve significantly reduced BER with optimal time offset to the collected driving data. We can also observe that the leading attack presents the higher BER on average. We think that this is due to slower reaction time and less precise velocity adjustment performed by the victim, as the victim has no intention to accurately synchronize vehicle velocity.

Meanwhile, we can determine the ECC that is required for a secure sketch in the reconciliation phase. The ECC should allow the legitimate devices to agree on the same key, however, it must not allow the adversary to recover the key. We observe that the maximum BER between the legitimate devices is 0.07, while the minimum BER at the attacker is 0.44 (i.e., an ECC that has error correction capability above 0.07 while below 0.44 satisfies the condition). Therefore, we select RS(63,37), which can correct up to 13 symbols among a codeword with 63 symbols (i.e., with error correction capability of 0.21).

Randomness of Generated Keys: It is crucial to guarantee that the established secret key is random since it would be used for encryption. To evaluate the randomness of the generated key, we run 8 randomness tests that are available in the NIST randomness test suite. The NIST test suite has 16 different statistical tests in total, however, as the bitstreams generated in this research satisfy the input requirement of the 8 tests only [24]. Each test returns a p-value as a result. To pass a test, p-value must exceed 0.01. Table I demonstrates the test results of a generated key, where the obtained p-values are all greater than 0.01 in all 8 tests.

V. Related Work

A. Traditional cryptographic secret key establishment

Traditional secret key establishment schemes (e.g., Diffie-Hellman key exchange [15]) are not practical solutions for mobile or IoT devices, as they require fixed key management structures, and mobile devices usually suffer from limited battery life and computational resources. Most importantly, the traditional schemes may suffer from usability issues for the large numbers of devices brought into a vehicle. Meanwhile, when a user pairs two devices using Bluetooth Secure Simple Pairing [27], the user is required to type a PIN or authorize the pairing process by responding "yes/no". However, the IoT devices may not always provide a user interface (e.g., touch screen, keypad), and the pairing procedure becomes cumbersome with multiple devices.

B. Context-based secret key establishment

The context-based secret key establishment schemes are proposed to overcome the aforementioned limitations of the traditional secure channel establishment techniques. They utilize the randomness embedded in the surrounding environment as a source of the secret key. As the user does not need to directly type PINs or confirm the pairing process, these schemes remove the requirement of human intervention for the device pairing process. A common idea of the context-based key establishment methods is that legitimate devices co-located within the same space may acquire highly similar sensor observations (e.g., ambient light and audio signal [23], acoustic signals [30], color shift observation over light channel [18], wireless signal strength [13] or wireless channel state information [19], [22]). However, these approaches share the following shortcomings: (1) they can be easily affected by environmental interferences (e.g., noise), (2) the user may need to inject artificial events to decrease pairing time. For example, a user's hand touch [33] for generating a channel between the wristband wearable device and the touched device, shaking or moving both devices for establishing the key by observing motion trajectories [17].

The IMU-based device pairing schemes for vehicular environments [6], [16] extract secret keys from the vehicle movement or environmental changes obtained from IMU sensor readings. However, they introduce two limitations: (1) As IMU sensor data can be easily interfered with by device movement or orientations, they require the legitimate devices to be fixed (e.g., taping the devices) inside of the vehicle. (2) Unlike permission-protected GPS data, IMU sensor data are considered insensitive, thus any mobile applications can obtain the data without explicit permissions [21], [35]. Instead, *GPSKey* utilizes GPS data to reduce impact of device movement/orientation. Our experiment results in Section IV-B show that a smartwatch on the driver's wrist obtains highly similar GPS data as the other legitimate devices.

VI. Conclusion

We propose a novel secret key establishment scheme, called *GPSKey*, for secure intra-vehicle communication. With *GPSKey*, the legitimate devices utilize highly similar GPS velocity observations in a moving vehicle to agree on the same key. The idea is to utilize the randomness embedded in GPS data as a source of the secret key. The daily driving scenarios present randomness from various uncontrollable factors, including traffic situations, and surrounding environments. We derive the secret key from GPS velocity data with enough randomness. Furthermore, we utilize a secure sketch to correct mismatching bits between the legitimate devices. Real-world experiment results show that legitimate devices can establish a 128-bit key in a 1.18-minute driving period, while an adversary following or leading the target vehicle fails to recover the legitimate devices' keys.

Acknowledgement

We would like to thank our anonymous reviewers for their insightful comments and feedback. This work was supported in part by the National Science Foundation under Grant No.1948547.

References

[1] N. H. T. S. Administration *et al.*, "Federal motor vehicle safety standards; v2v communications," *Federal Register*, vol. 82, no. 8, pp. 3854–4019, 2017.

[2] E. G. N. S. S. Agency, "Gsa gnss market report, issue 6," https://www.euspa.europa.eu/system/files/reports/market_report_issue_6_v2.pdf, 2019.

[3] B. Brecht and T. Hehn, "A security credential management system for v2x communications," in *Connected Vehicles*. Springer, 2019, pp. 83–115.

[4] L. Buttyán, T. Holczer, A. Weimerskirch, and W. Whyte, "Slow: A practical pseudonym changing scheme for location privacy in vanets," in *2009 IEEE Vehicular Networking Conference (VNC)*, 2009, pp. 1–8.

[5] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, 2008.

[6] M. Fomichev, J. Hesse, L. Almon, T. Lippert, J. Han, and M. Hollick, "Fastzip: Faster and more secure zero-interaction pairing," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, p. 440–452.

[7] M. Fomichev, F. Álvarez, D. Steinmetzer, P. Gardner-Stephen, and M. Hollick, "Survey and systematization of secure device pairing," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 517–550, 2018.

[8] A. Ghosal and M. Conti, "Security issues and challenges in v2x: A survey," *Computer Networks*, vol. 169, p. 107093, 2020.

[9] M. T. Goodrich, M. Sirivianos, J. Solis, C. Soriente, G. Tsudik, and E. Uzun, "Using audio in secure device pairing," *Int. J. Secur. Netw.*, vol. 4, no. 1/2, pp. 57–68, 2009.

[10] U. S. C. Guard, "Navstar GPS User Equipment," https://www.hsdl.org/?view&did=22298, 1996.

[11] J. Han, A. Chung, M. Sinha, M. Harishankar, S. Pan, H. Y. Noh, P. Zhang, and P. Tague, "Do you feel what i hear? enabling autonomous iot device pairing using different sensor types," 05 2018, pp. 836–852.

[12] Hongbo Liu, Jie Yang, Yan Wang, and Yingying Chen, "Collaborative secret key extraction leveraging received signal strength in mobile wireless networks," in *IEEE INFOCOM*, 2012, pp. 927–935.

[13] 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 *ACM MobiCom*, 2009.

[14] N. Karapanos, C. Marforio, C. Soriente, and S. Capkun, "Sound-proof: Usable two-factor authentication based on ambient sound," in *USENIX Security*, 2015, pp. 483–498.

[15] C. Kaufman, R. Perlman, and M. Speciner, *Network security : private communication in a public world*, ser. Prentice Hall series in computer networking and distributed systems, 2002.

[16] K. Lee, N. Klingensmith, D. He, S. Banerjee, and Y. Kim, "ivpair: context-based fast intra-vehicle device pairing for secure wireless connectivity," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 25–30.

[17] Z. Li, Q. Pei, I. Markwood, Y. Liu, and H. Zhu, "Secret key establishment via rss trajectory matching between wearable devices," *IEEE Trans. on Info. Forensics and Security*, vol. 13, no. 3, pp. 802–817, 2018.

[18] H. Liu, B. Liu, C. Shi, and Y. Chen, "Secret key distribution leveraging color shift over visible light channel," in *IEEE Conf. on Communications and Network Security (CNS)*, 2017, pp. 1–9.

[19] H. Liu, Y. Wang, J. Yang, and Y. Chen, "Fast and practical secret key extraction by exploiting channel response," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 3048–3056.

[20] Y. Lu, F. Wu, S. Tang, L. Kong, and G. Chen, "Free: A fast and robust key extraction mechanism via inaudible acoustic signal," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 311–320.

[21] A. Maiti, M. Jadliwala, J. He, and I. Bilogrevic, "Side-channel inference attacks on mobile keypads using smartwatches," *IEEE Trans. on Mobile Computing*, vol. 17, no. 9, pp. 2180–2194, 2018.

[22] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, "Radio-telepathy: Extracting a secret key from an unauthenticated wireless channel," in *ACM MobiCom*, 2008, pp. 128–139.

[23] M. Miettinen, N. Asokan, T. D. Nguyen, A.-R. Sadeghi, and M. Sobhani, "Context-based zero-interaction pairing and key evolution for advanced personal devices," in *ACM CCS*, 2014, pp. 880–891.

[24] N. I. of Standards & Technology, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," Tech. Rep., 2010.

[25] G. Oligeri, S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro, "Drive me not: Gps spoofing detection via cellular network: (architectures, models, and experiments)," in *ACM WiSec*, 2019, pp. 12–22.

[26] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & Systems (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[27] J. Padgette, J. Bahr, M. Batra, M. Holtmann, R. Smithbey, L. Chen, and K. Scarfone, "Guide to bluetooth security," *NIST Special Publication*, vol. 800, p. 121, 2017.

[28] 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*, vol. 9, no. 1, pp. 17–30, 2009.

[29] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel: Design and usability study," *IEEE Trans. on Info. Forensics and Security*, vol. 6, no. 1, pp. 28–38, 2011.

[30] D. Schürmann and S. Sigg, "Secure communication based on ambient audio," *IEEE Trans. on Mobile Computing*, 2013.

[31] N. Tippenhauer, C. Pöpper, K. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *ACM CCS*, 2011.

[32] U-blox, "Neo-6 series," https://www.u-blox.com/en/product/neo-6-series, 2021.

[33] W. Wang, L. Yang, and Q. Zhang, "Resonance-based secure pairing for wearables," *IEEE Trans. on Mobile Computing*, vol. 17, no. 11, pp. 2607–2618, 2018.

[34] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems," in *USENIX Security*, 2018, pp. 1527–1544.

[35] S. Zhuo, L. Sherlock, G. Dobbie, Y. S. Koh, G. Russello, and D. Lottridge, "Real-time smartphone activity classification using inertial sensors—recognition of scrolling, typing, and watching videos while sitting or walking," *Sensors*, vol. 20, no. 3, p. 655, 2020.