DASK: Driving-Assisted Secret Key Establishment

Edwin Yang*, Song Fang*, and Dakun Shen†
*University of Oklahoma, Norman, OK, 73019, {edwiny, songf}@ou.edu
†Central Michigan University, Mount Pleasant, MI, 48859, shen2d@cmich.edu

2022 IEEE Conference on Communications and Network Security (CNS) | 978-1-6654-6255-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/CNS56114.2022.9947241

***Abstract*—Low-cost and easily obtained Global Navigation Satellite System (e.g., GPS) receivers are broadly embedded into various devices for providing location information. In this work, we develop a secret key establishment by utilizing the driving data obtained from GPS. Those data may exhibit randomness as the driver may alternatively step on the accelerator and brake pedals from time to time with varying force in order to adapt to the road traffic during driving. A driving vehicle provides a physically secure boundary as the devices co-located within the vehicle can observe common GPS data, as opposed to devices that do not experience the trip. We implement this key establishment in a real-world environment on top of off-the-shelf GPS-equipped devices as well as widely deployed GPS modules each connected with Raspberry Pi. Extensive experimental results show that when a user drives around 1.36 km for 1.32 minutes on average under moderate traffic conditions, two legitimate GPS-equipped devices in the vehicle can successfully establish a 128-bit secret key. Meanwhile, an attacker following the target vehicle is unable to establish a secret key with the legitimate devices.**

I. Introduction

Global Navigation Satellite System (GNSS) has become a ubiquitous technology and has been built into connected vehicles and various other devices (e.g., smartphones, tablets, or IoT devices). The data those devices collect often carry a great potential of privacy risks in relation to the use of the data and its access [1]–[3]. The popularity of GNSS-equipped mobile devices demands a pairing scheme, so that each pair of devices, which do not share prior knowledge with each other, can communicate securely. This challenge has not yet been fully addressed [1], [3].

On the other hand, with the advent of in-vehicle infotainment (IVI) systems (such as Android Automotive [4]) and other mobile devices with limited user interfaces (e.g., smartwatches), there is an increasing need for a secure and spontaneous pairing technique. In general, such a device pairing method should satisfy the following four requirements.

- *Lightweight*: some devices are cheap and may not afford complex computation, and thus the method must not require complex and expensive hardware;
- *Zero-effort*: user interfaces are not always available, so it may be unrealistic to require user involvement (e.g., typing a password or bringing both devices close to communicate via a Near Field Communication channel), and also human involvement may cause usability concerns;
- *Flexible*: considering the scalability and portability of such GPS-equipped devices, it should enable quick key establishment and update;
- *Secure*: it should resist man-in-the-middle (MITM) attacks, which are the common challenges in secure device pairing [1], [5] in the absence of pre-authentication of each encountered device.

Traditional cryptographic methods (e.g., Diffie-Hellman key exchange) rely on computational hardness and are neither lightweight nor flexible. Also, they often need user intervention for authentication operations to prevent MITM attacks. For example, one device generates a passkey and displays it to the user, who is then required to type it into the other device [6]. The devices finally run a shared secret authentication protocol which succeeds if the input passkey is correct. The attacker's connection request would be rejected without the correct passkey. There is often a one-time inconvenient paring phase when the user needs to enter a passkey shown in one device into another. After that, the pairing would be automatic by reusing the pre-negotiated passkey, which, however, can be vulnerable to numerous attacks [7].

Instead, researchers have proposed various context-based pairing mechanisms which can mitigate human involvement via the common observation, e.g., a visual channel [8], ambient audio patterns [9], [10], event timing [2], radio frequency noise [11], wireless signal strength [12], [13] and wireless channel [14], [15]. The uncertainty in the observed context generates the randomness and thus provides the entropy for the established key. However, they impart two major drawbacks.

First, the randomness embedded in those contexts is not always enough to generate effective keys, leading that the users may have to deliberately inject extra contexts. For example, [9] requires the user to generate some noise in order to work in a quiet environment while [2] needs the user to introduce extra events to decrease the key generation time in a quiet house. Such deliberate solutions obviously hinder the practicality in terms of cost and usability [2].

Second, as there is no authentication of each party, most existing work [10], [12]–[14] are subject to the MITM attack, in which an adversary stealthily relays and possibly alters the communication between two parties by establishing two distinct keys, one with each party. Authors in [2] propose a secure pairing technique that is resistant to MITM attacks by utilizing common inter-event timing information, while this technique targets the scenario when both devices are within a physical home and the pairing time is inversely proportional to the frequency of events.

In this paper, we develop a practical driving-assisted secret key establishment, called *DASK*, with the aforementioned four important features. The basic idea is to leverage the fact that two GNSS-equipped devices in the same vehicle can capture

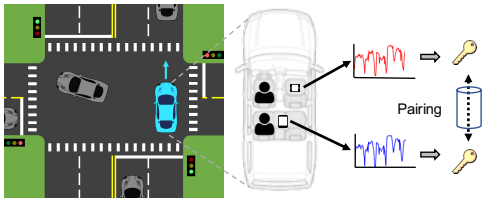

Fig. 1. GPS-equipped devices can use driving data for key establishment.

the same driving data and thus extract the same secret keys from them. Specifically, each device quantizes the data into initial binary bit sequences. Due to observation errors, there may exist mismatched bits between the initial binary sequences. Reconciliation is then applied to make both devices agree on an identical secret key, with which the two devices can do secure communication. For example, DASK can be used to connect a mobile device to an in-vehicle infotainment (IVI) system, while the current user-involved pairing process is often tedious and unsafe to do while driving [16].

DASK is lightweight since GNSS receivers are inexpensive and widely available across various devices. Second, it enables devices to automatically establish a secret key using the observed driving data without human involvement. Also, daily driving normally embeds rich randomness in nature so that humans are not required to generate deliberate contexts. Third, it is flexible as each pair of devices put in the same driving vehicle may launch the pairing and the established key can be updated with a new driving trip. Also, only the devices experiencing the same driving period for the key generation can pair with each other. Thus, an attacker who does not locate in the same vehicle with legitimate devices when key generation is performed will not be able to extract the established key. A vehicle provides a physically secure boundary and MITM attacks can be effectively prevented.

Our design is based on two key insights. On one hand, a driver may alternately step on or release the accelerator and brake pedals with varying force in order to adapt to the road traffic. Consequently, the vehicle may switch among different states, i.e., accelerated, decelerated, and uniform motions. Such states can be captured by in-vehicle GNSS-equipped devices. Meanwhile, the road traffic is normally dynamic, diverse, and random, so is the vehicle's state transition sequence. Thus, the inherent randomness in the vehicle's states during driving can be utilized as a source for extracting a secret key between these devices, as shown in Figure 1.

On the other hand, a following vehicle may track the trajectory of the target (preceding) vehicle. However, it is almost impossible for a stalker (i.e., the driver in the following vehicle) to calculate or copy exactly each driving behavior of the preceding driver, considering the line of sight blocking introduced by unpredictable traffic or complex roadside environment. Especially under certain crowded traffic slots, there may generate a lot of minute state transitions that cannot be repeated by the stalker. For example, the driver in the target vehicle steps on the brake pedal lightly to decelerate a little bit, and then soon switch to stepping on the accelerator pedal to accelerate. Thus, the following car may miss those minute state transition details. With the driving data that is different from the target user, the stalker would generate a key different from the key established between the devices in the target vehicle.

Some recent studies propose to establish keys with inertial measurement unit (IMU) sensors for intra-vehicle devices [16], [17]. However, *DASK* has two advantages over them.

First, accelerometer data can be easily disturbed by device orientation/motion [16], [17] while GPS is less affected by those factors. Specifically, to eliminate the effect of device movement, [16] uses an adhesive tape to fix the devices while [17] strictly fixes the devices with belts. Also, [16] requires that the legitimate devices in the vehicle must be placed in the same orientation. To relax the requirement of device orientation consistency, [17] proposes a calibration method by combining accelerometer and gyroscope data. However, an extra sensor (i.e., gyroscope) is required. Any device movement may affect the accelerometer and gyroscope readings, leading to failure of the key establishment, as pointed out by [17].

Second, unlike permission-protected sensors (e.g., GPS, microphone, and camera), IMU sensors are normally considered insensitive and accessible by any mobile application without special privilege [18], [19], while applications' permission to access GPS is user-manageable and thus it would be more secure to utilize GPS data to generate secret keys.

Our main contributions are summarized as follows.

- We build a customized method to separate useful driving data that exhibit randomness and investigate the practical application of using such data in key generation.
- Compared with IMU sensor based secret key establishment schemes, *DASK* is more robust to device deployment and movement, and more secure by employing permission-protected location data as the key source.
- Our real-world experiments show with moderate driving time/distance, *DASK* enables various GPS-equipped devices to establish a secret key, demonstrating the feasibility, effectiveness, and robustness.
- *DASK* is resilient to MITM attacks as an attacker without experiencing the same driving trip of key establishment with legitimate devices is unable to extract the secret key.

II. Adversary Model

In a general scenario, Alice and Bob aim to establish a secret key. Either party could be any GPS-equipped device brought into the vehicle. The utilized vehicle is a standard family car with width, length, and height of about 1.8, 4.8, and 1.5 meters, respectively. We assume that no adversary appears in the car during the key establishment period. This assumption follows the basic rule for all context-based key establishment schemes (e.g., [2], [16], [17], [20]). For example, [2] utilizes the physical boundaries of a house to separate legitimate devices and adversaries. Also, it is easy to detect an active GPS device with a readily available GPS Bug Detector. For a passive one, as the recording cannot be accessed until the attacker gets the tracker back after some time, it would be difficult for the

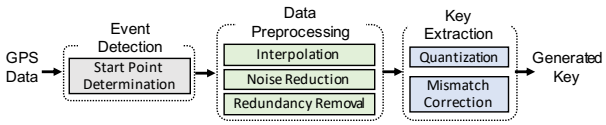

Fig. 2. *DASK* scheme flow chart.

attacker to determine effective data for the key establishment from a huge amount of recorded data.

We consider that an attacker, Eve, can launch non-invasive secret key inference attacks by stalking the target vehicle. Due to the line-of-sight obstruction of various traffic factors, Eve is unable to directly observe the driving movement of the preceding vehicle for the whole key generation period. Note that this process may require dedicated speed measurement systems, also if Eve follows the target vehicle too close for a long time, the exposure risk would be dramatically increased. Instead, the attacker takes another intuitive strategy, i.e., using her observed GPS data to infer the key. Moreover, Eve has knowledge of the applied secret key establishment algorithm.

III. System Design

A. Scheme Outline

Alice and Bob, co-located within a vehicle, both continuously capture GPS when the vehicle runs. In this work, we focus on the velocity information as the time-derivative of its gathered position data. The velocity variation shows the driving vehicle states, which are the results of pedal controls performed by the driver. With recorded GPS, each device derives the velocity as well as the corresponding timestamp information. To launch *DASK*, both devices first pre-agree on a start point of the key generation. Then, each performs a data pre-processing phase in which the noise and bias in the GPS measurements, as well as the redundant driving data (i.e., a long-time stop or cruise) without exhibiting randomness, will be eliminated. Next, each device computes a binary bit sequence from the pre-processed data with the customized quantization method. Finally, a reconciliation process is performed to correct mismatches between the generated sequences on both sides. Figure 2 shows the flow chart of *DASK*.

B. Event Detection

1) GPS Accuracy: The accuracy of GPS observations depends on multiple factors, including the satellites' positions, surrounding landscape, and receiver hardware characteristics [21]. For Alice and Bob within a normal-sized vehicle, the effects of the first two factors are almost the same. GPS requires a direct line of sight between the receiver and the satellite. Its accuracy suffers due to reflections and attenuations, especially in urban environments with dense obstacles such as high buildings and trees. However, as the distance between Alice and Bob is much smaller compared with the length of the signal path from the satellite to the receiver, the fluctuation of GPS accuracy is consistent across both devices.

On the contrary, the hardware difference between GPS receivers (e.g., frequency band, antenna, and positioning algorithm) may bring disparities between their respective records. Intuitively, a low-cost receiver may be insensitive to weak GPS signals and thus have more errors, while a high-end dual frequency, the survey-grade receiver would get a more accurate measurement. In Section IV-A, we test various off-the-shelf GPS-equipped devices and widely deployed GPS receiver modules to explore the measurement differences.

2) Synchronization of Start Points: DASK does not immediately initialize the key establishment when the vehicle starts, and instead waits until the vehicle runs for a while. The period from the ignition to the start point of key establishment is called the preparation phase, which aims to make the key establishment efficient considering that the vehicle state may be easily observed and shows less frequent variation during this phase. Each device then records the candidate start point, i.e., the time when the observed vehicle speed firstly reaches a threshold of V_s, which will be set based on the observation that the vehicle state varies gradually more frequently afterward.

Due to GPS measurement errors or signal delay, the candidate start points identified by Alice and Bob may not be exactly the same. To solve this problem, *DASK* enables one party (e.g., who launches the key establishment) to notify the other party of the start point with the corresponding GPS timestamp which can be sent simultaneously with other information during the reconciliation, as discussed in Section III-D. The GPS timestamps are accurate and synchronized as they originate from each satellite's high precision atomic clocks, which are periodically corrected by the Master Control Station [22], i.e., the central control node for the GPS satellite constellation. Both devices then select a same timestamp as the common start point of the key establishment.

C. Data Pre-processing

1) Interpolation: There is no guarantee that Alice and Bob always successfully obtain GPS signals due to the dynamic variation of the environment outside the vehicle, and each device sometimes receives empty GPS data (i.e., null values). Therefore, GPS data should be interpolated, and we utilize linear interpolation to estimate the missing velocity values.

2) Noise Reduction: It is observed that the frequencies of the variations in GPS velocity due to vehicle control primarily lie at low frequencies. To preserve valuable signal components and mitigate the random high-frequency noise introduced by environmental interference or hardware imperfection in GPS velocity data, we thus adopt a Butterworth low-pass filter [23].

3) Redundancy Removal: As mentioned earlier, when the vehicle stops or moves at a constant speed with cruise control, the observed velocity would be stable and easily discovered by out-of-the-vehicle observers (i.e., the velocity of the vehicle may not provide sufficient randomness).

Stop Period Detection: When the vehicle stops, the obtained velocity may slightly deviate from 0 due to the noise or unexpected GPS signal lagging. Let v_t and p_t denote the pre-determined maximum measurement errors of speed and coordinate, respectively. When the vehicle stops or moves at a speed less than v_t, the GPS coordinate usually maintains a

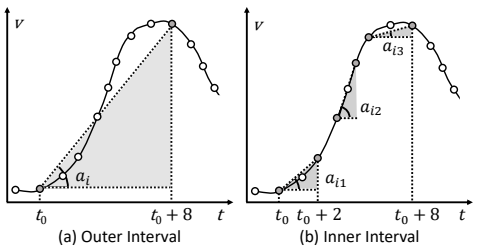

Fig. 3. An example of an outer interval and corresponding inner intervals: the acceleration of the outer interval is a_i, and the acceleration values of the three inner intervals are denoted with a_{i1}, a_{i2} and a_{i3}. As $a_{i1} < a_i$, $a_{i2} > a_i$, and $a_{i3} < a_i$, the three inner intervals are quantized into "0", "1" and "0".

consistent value (i.e., the variation p_t is very small). We thus utilize such a phenomenon to distinguish stop periods.

Cruise Period Detection: When a vehicle is in cruise mode, the observed velocity normally varies significantly small among consecutive samples. The maximum speed variance is set as V_{max}, which depends on the velocity fluctuation during cruise and can be pre-obtained through an empirical profile. Let $\mathbf{v} = [v_1, v_2, \cdots, v_N]$ denote the N-sample velocity sequence after noise reduction. We utilize a sliding window method to detect a set C of cruise periods each with no less than w_0 successive samples. Initially, let $C = \emptyset$, the current window size $w = w_0$ and the index $i = 1$, and we then iteratively do the following steps.

(a) From $\mathbf{v}$, obtain the candidate cruise period, $\mathbf{c} = [v_i, \cdots, v_{i-1+w}]$.
(b) Calculate the speed variance $\sigma^2 = \frac{1}{w}\sum_{j=i}^{i-1+w}(v_j - \bar{v})^2$ over the candidate cruise period, where $\bar{v} = \frac{1}{w}\sum_{j=i}^{i-1+w} v_j$.
(c) If $\sigma^2 < V_{max}$, $w = w + 1$ and jump to step (a).
(d) If $w > w_0$, a cruise period is detected and we have $C = C \cup \{\mathbf{c}\}$ and $i = i + w$, otherwise, we reset $i = i + 1$; jump to step (a) until all samples are processed.

With the detected stop and cruise periods, each party will then remove them. Let $R = [(t_{s_1}, t_{e_1}), \cdots, (t_{s_r}, t_{e_r})]$ denote the identified r-segment redundancy time information, where t_{s_i}, t_{e_i} $(i \in \{1, \cdots, r\})$ represent the start and end time of every segment respectively. Though the detected periods between legitimate devices are quite similar empirically, they may not perfectly match. To eliminate the inconsistencies, an intuitive method is to enable both legitimate devices to share detected redundancy time information. However, redundancy information exchange will introduce extra costs and slow the key establishment. To increase the efficiency of key establishment, *DASK* only requires one party to detect the redundancy and deliver such information to the other party so that both can filter out the same redundant periods.

D. Key Extraction

DASK develops a customized quantization method to enable each device to generate an initial binary sequence based on the pre-processed velocity data. As the sequences generated at both devices after quantization may not perfectly match due to various measurement errors, the technique of secure sketch [24] is then utilized to correct those mismatches.

1) Quantization: To maintain the stability and fuel efficiency of a vehicle, a driver usually pivots the foot to the brake/gas pedal accordingly and adjusts the vehicle state every a moderate time (e.g., several seconds). For traditional value based quantization methods with a single threshold, a sample value above a pre-determined threshold will be encoded as 1 while the rest will be encoded as 0. If such a quantization method is enforced, it may frequently generate a series of all 1 or all 0 sequences. Thus, they are not appropriate for extracting randomness from the GPS velocity stream. On the other hand, for a multi-level quantizer, it is difficult to determine the optimal number of quantization thresholds, and also more levels may make the key extraction more vulnerable to measurement errors.

Algorithm 1 Two-level Interval Based Quantization

Input: A velocity stream $\mathbf{v}$ with N samples; outer and inner interval sizes s_o and s_{in}
Output: A binary bit sequence K_Q

$K_Q \leftarrow \emptyset$, $d \leftarrow \lfloor N/(s_o - 1) \rfloor$, $n \leftarrow \lfloor s_o/s_{in} \rfloor$
for $i = 1$ *to* d **do**
  $O_{end} = i * s_o$, $O_{start} = O_{end} - (s_o - 1)$
  $a_i \leftarrow \left(\frac{v[O_{end}] - v[O_{start}]}{s_o - 1}\right)$ # outer interval
  for $k = 1$ *to* n **do**
    $I_{end} = O_{start} + k * s_{in} - 1$, $I_{start} = I_{end} - (s_{in} - 1)$
    $a_{ik} \leftarrow \left(\frac{v[I_{end}] - v[I_{start}]}{s_{in} - 1}\right)$ # inner interval
    if $a_{ik} > a_i$ **then**
      $K_Q \leftarrow [K_Q;\ 1]$
    else
      $K_Q \leftarrow [K_Q;\ 0]$
    end if
  end for
end for

Instead of quantizing the velocities one by one with fixed thresholds, we develop a two-level interval based quantization method, which comes from the observation that a driver usually utilizes non-uniform force to operate the pedals, leading that the acceleration varies even during a small period. Specifically, the proposed method takes two steps: 1) *Division*: divide the whole velocity stream into consecutive sub-sequences, called outer intervals, and split each outer interval into non-overlapping sub-subsequences, called inner intervals; and 2) *Reconstruction*: calculate the acceleration values of each outer interval and corresponding inner intervals, and each inner interval is encoded by 1 if its acceleration value is larger than that of its corresponding outer interval while otherwise 0. Figure 3 shows an example of the proposed two-level interval based quantization method with a 9-sample outer interval and 3-sample inner interval. As a result, each outer interval with n inner intervals can generate n bits. Algorithm 1 presents the pseudocode for the proposed quantization method.

The selection of interval sizes is a tradeoff between the key length and the randomness of the generated bitstream. The inner interval size must be smaller than the outer interval size. Also, the total number of inner intervals is equal to the length of

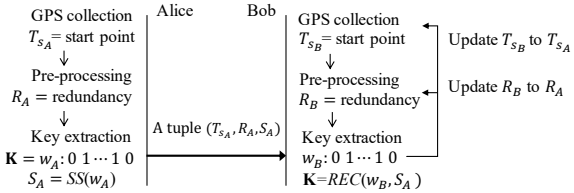

Fig. 4. Timing diagram for *DASK*.

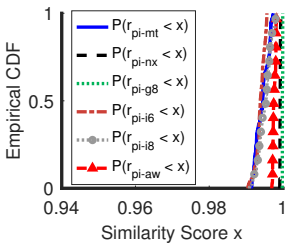

Fig. 5. CDFs of cross-correlation.

the quantized bits. Meanwhile, a small inner interval size may bring a series of all 1 or all 0 sequences when the vehicle state changes slowly. In general, the selection of the interval sizes may adjust based on the specific driving situation. Both legitimate devices can utilize a heuristic method to determine the interval size, i.e., gradually increase the length of each interval size until a secret key is obtained that satisfies both the length and randomness requirements.

2) Mismatch Correction: Ideally, if two devices observe exactly the same GPS velocities, they would obtain the same quantization outputs. However, due to hardware differences and random noise, GPS velocity records at both devices may result in a small number of mismatched bits between the two generated binary bit sequences. Therefore, we use a secure sketch to make both devices agree on an identical secret key.

Specifically, with the input sequence, Alice calculates the parity bits according to a selected ECC rule to obtain a sketch, which can be then made public. Take a Reed-Solomon (RS) ECC $RS(n,\ k)$ as an example, which has n symbols of s bits each. The first k of the n symbols are called information bits, and the rest parity bits are calculated based on the RS algorithm and the information bits. That means, for a sequence with $k*s$ bits, the calculated sketch has $(n-k)*s$ bits. Given a symbol size s, the maximum length of the codeword C is $m = 2^s - 1$, so $n \leq m$ should hold. Correspondingly, the RS decoder can correct any $\frac{n-k}{2}$ symbol errors in the codeword.

Figure 4 presents the timing diagram for *DASK*. Alice generates the secret key bits $\mathbf{K}$ and then publishes a tuple of information (i.e., start point timestamp T_{S_A}, start and end time of each detected redundant segment R_A, and the sketch S_A) to launch the key establishment. Note that which party to send the tuple (i.e., launch the key establishment) may depend on the agreement between Alice and Bob or who firstly generates the required key length. On the other hand, with T_{S_A} and R_A, Bob updates corresponding values and makes the pre-processed velocity stream consistent with that of Alice in time. Bob then utilizes a recovery procedure to obtain the secret key. Once the length of the quantized bits reaches the required length of the information bits of a chosen ECC, the recovery procedure combines them with the received sketch (i.e., parity bits) to obtain an ECC codeword and decode it, and the information bits in the decoded codeword are recovered key (i.e., $\mathbf{K}$).

E. Security Analysis

1) Passive Attack: In general, a passive adversary cannot directly observe or measure the minute state transitions of the target vehicle for a long time from outside due to various obstacles that block the direct line-of-sight path. Instead, she may attempt to launch the *stalking attack*: trying to maintain the same driving state by following the target vehicle to make her collected GPS data similar to that utilized for key generation between legitimate devices. We explore the impact of such an attack in Sections IV-D and IV-E.

We further consider a stronger passive attack, i.e., the attacker can measure a target vehicle's velocity by installing multiple cameras on the road, utilizing a drone with a camera, or employing a speed radar gun. However, we note that legitimate devices in the same vehicle experience highly similar GPS errors (that contribute to the derived key) while the attacker cannot obtain/control it. We demonstrate the impact of the speed measurement attack in Section IV-D.

2) Active Attack: GPS is vulnerable to spoofing attacks where adversaries may try to inject falsified GPS signals over the air to the victim's GPS receivers [25], [26]. However, to launch such attacks, the adversaries must carefully select spoofed routes to make them consistent with the physical road network, otherwise, there often exists a noticeable contradiction between the vehicle's actual routing on physical roads and the corresponding GPS driving record [26]. Meanwhile, a GPS receiver may cross-check the collected GPS data with dead reckoning results based on IMU sensors (e.g., accelerometer) [26] or with the messages received from the mobile cellular network [27] to detect GPS spoofing attacks.

IV. Experimental Evaluation

Our prototype consists of two legitimate devices and an adversary device. Each could be an Android device (LG Nexus 5X or LG G8X smartphone/Lenovo Moto Tab tablet), an iOS smartphone (iPhone 6S+/8), a smartwatch (Apple Watch Series 3), or a Raspberry Pi 3 connected with a NEO-6M module [28]. The user puts two legitimate devices (Dev_A and Dev_B) in a vehicle, and they launch *DASK* with the collected GPS time series when the vehicle moves. The attacker drives another vehicle with the adversary device (Dev_E) and follows the user's vehicle. We use the following metrics:

- **Bit generation rate (BGR)**: This is the number of generated secret bits per second, using the GPS data collected in a driving duration.
- **Bit mismatch rate (BMR)**: This is the percentage of bits in disagreement between the initial binary bit sequences after the quantization process at the two devices.
- **Randomness**: We measure the key randomness with the standard NIST randomness test suite [29].

A. Device Impact

Cross-correlation is widely used to measure the *similarity* between two time series. Let $\mathbf{v}_a = [v_{a_i}, \cdots, v_{a_{i+T-1}}]$ and $\mathbf{v}_b = [v_{b_i}, \cdots, v_{b_{i+T-1}}]$ denote two collected T samples of normalized GPS velocities. The cross-correlation $R_{\mathbf{v}_a\mathbf{v}_b}(l)$ can be calculated by a function of the lag $l \in [0, T-1]$ applied to $\mathbf{v}_b$, i.e., $\sum_{j=1}^{T} v_{a_{i+j-1}} \cdot v_{b_{i+j-1-l}}$, where $v_{b_i} = 0$ if $i \leq 0$. To accommodate for different amplitudes of the two series, the cross-correlation can be normalized as $R'_{\mathbf{v}_a\mathbf{v}_b}(l) = \frac{R_{\mathbf{v}_a\mathbf{v}_b}(l)}{\sqrt{R_{\mathbf{v}_a\mathbf{v}_a}(0) \cdot R_{\mathbf{v}_b\mathbf{v}_b}(0)}}$, which lies in the range of $[-1, 1]$ at lag l, with 1 indicating perfect correlation, -1 denoting perfect anticorrelation, and 0 showing uncorrelation. We derive the largest absolute value of cross-correlation, $\max_l(|R'_{\mathbf{v}_a\mathbf{v}_b}(l)|)$ to quantify the similarity between two velocity streams.

We enable the user to wear Apple Watch 3, referred to as aw, and place the other six test devices in the same vehicle, including Lenovo Moto Tab, LG Nexus 5X, LG G8X, iPhone 6S+, iPhone 8, and a Raspberry Pi 3 connected with a NEO-6M GPS module, referred to as mt, nx, g8, i6, i8, and Pi respectively. All devices collect the driving data simultaneously. We repeat the experiment 100 times, each with 180 GPS samples. Let r_{Pi-var} ($var \in$ {mt, nx, g8, i6, i8, aw}) denote the calculated cross-correlation between the velocity sequences collected by the Raspberry Pi and corresponding one of the other devices. Figure 5 shows the empirical cumulative distribution functions (CDFs) of each respective cross-correlation value. We can see that all cross-correlation values reach above 0.9911 with a probability of at least 95.26%, implying that acquired GPS data from different devices are highly consistent. It is worth mentioning that though the smartwatch frequently moves as the user has to operate the steering wheel, its recorded GPS data are still highly similar to those recorded by the rest devices.

B. Trip-dependent Uniqueness of GPS Data

We enable a target user to drive on the same typical urban road 100 times, and a stalker driving another car follows the target user. The legitimate devices and the adversary device all collect GPS data for each driving trip.

As an example, Figure 6 shows the obtained GPS velocities by legitimate devices as well as the adversary device. First, we observe that the velocity fluctuates from time to time. Next, we see that the velocities of two legitimate devices are quite similar while both significantly deviate from that of the adversary device. Besides, starting from the 80^{th} to 130^{th} second, the adversary has to wait for the traffic light while the target user keeps moving. Such a situation further increases the differences between their respective velocity observations.

Figure 7 plots the CDFs of the cross-correlation r_{ab} between the velocity sequences of Dev_A and Dev_B, r_{ae} between that of the Dev_E and Dev_A, as well as r_{self} between the velocity sequences obtained by the same device for two different driving trips on the same road. We can observe that r_{self} is less than 0.47 with a probability of 0.95, meaning that each driving trip on the same road would generate quite a different velocity stream. Also, there is a clear gap between

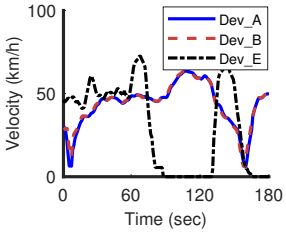

Fig. 6. Observed velocities.

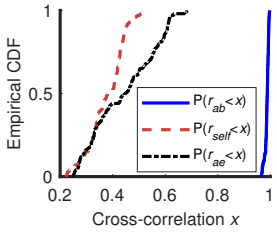

Fig. 7. CDFs of r_{ab}, r_{self} and r_{ae}.

r_{ae} and r_{ab}, demonstrating that legitimate devices in the target vehicle will always observe similar velocity streams while the stalker in the following car would obtain a different one.

C. Quantization Interval Selection

With the two-level interval based quantization method, the selection of the inner and outer interval sizes (i.e., s_{in} and s_o) is important as they are related to the quantization error, bit generation rate, as well as the randomness of the generated bit sequence. Meanwhile, to secure private communication, the established key should have a suitable length. We observe when $s_o \geq 10$, the quantized bit stream often has a series of all 1 or all 0 sequences. Considering both the key generation speed and avoiding low randomness in the generated bit stream, we enable $s_o = 9$. As $s_{in} < s_o$, we then vary s_{in} from 2 to 8 in increments of 1. For each s_{in}, we perform 100 attempts of quantizing a bit stream of length L ($L \in \{128, 160, 192\}$).

Figure 8 shows the average bit mismatch rate between two generated binary bit sequences at two legitimate devices when different interval value is utilized. We have the following three observations. First, the bit mismatch rate is always below 0.09 regardless of the interval size. Next, the bit mismatch rate varies slightly for different key lengths. This consistency demonstrates that the mismatched bits are almost uniformly distributed over the time series of binary sequences. Finally, the bit mismatch rate gradually increases when s_{in} increases until $s_{in} = 4$ and maintains stable when $s_{in} \geq 4$. This is because when s_{in} is closer to s_o, the acceleration of the inner interval is closer to that of the outer interval, causing the quantization result to be more easily affected by interference.

Figure 9 shows the average bit generation rate. We observe that the bit generation rate decreases with the interval size increasing from 2 to 5 while the bit generation rate for $s_{in} \geq 5$ remains unchanged. This is because each 9-sample outer interval will be converted into a binary sequence of length $\lfloor 9/s_{in} \rfloor$, which decreases when s_{in} increases from 2 to 5 while does not change when $s_{in} \geq 5$. Also, the actual bit generation rate is often less than its ideal maximum value (e.g., $\frac{\lfloor s_o/s_{in} \rfloor}{s_o} \cdot f_s$, where f_s is the GPS sampling rate) which can be obtained when the collected data have no redundancy. Besides, we can see that no matter which interval size is used, the bit generation rate maintains consistency for different key lengths.

D. Passive Attack Resistance

Stalking Attack: We measure the bit mismatch rate between the binary bit sequences after quantization. Figure 10 shows

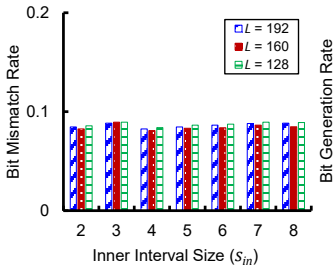

Fig. 8. BMR vs. s_{in} (legitimate).

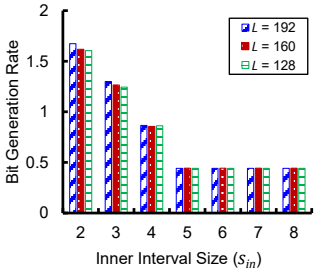

Fig. 9. BGR vs. s_{in}(legitimate).

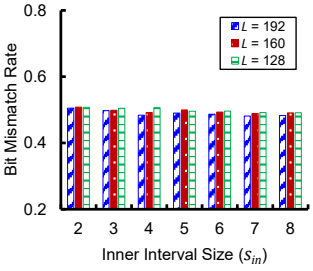

Fig. 10. BMR (stalking attack).

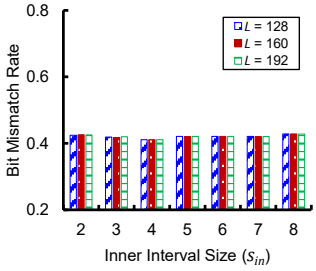

Fig. 11. BMR (measurement attack).

TABLE I
p-VALUE OF NIST TEST RESULT

NIST Test	p-value	NIST Test	p-value
Frequency	0.8597	Cummulative Sums	0.7375
Block Frequency	0.8415	Runs	0.0132
DFT	0.8711	Serial	0.0129
Longest Run	0.1671	Approximate Entropy	0.0885

BMR at the adversary, where BMR is always above 0.48, showing *DASK* can resist the stalking attack.

Speed Measurement Attack: We let an attacker continuously measure a target vehicle's velocity at a shoulder of a road with a speed radar gun (BUSHNELL Velocity Speed Gun, Model No.101911 [30]). We repeat the experiment 100 times. We see BMR at the attacker in Figure 11. First, the attacker achieves slightly decreased BMR than in stalking attack scenarios due to more accurate velocity measurements. The adversary's BMR is always above 0.41, which is clearly higher than the legitimate devices' BMR ($\leq$ 0.09). This gap lets the benign devices select an effective ECC that prevents the attacker from obtaining the legitimate key.

E. Key Generation

After quantization, we utilize an error correction code (ECC) based secure sketch to enable legitimate devices to agree on an identical key. The criteria of a qualified ECC is that legitimate devices can correct the bit discrepancies while the adversary cannot recover the same key with the help of the ECC. We define the *error correction capability* of an ECC as the ratio of the maximum number of erroneous bits that can be recovered to the total number of the bits in a codeword. For an example, for a $RS(n,\ k)$ ECC, its error correction capability is $\lfloor \frac{n-k}{2} \rfloor$. Therefore, the error correction capability of the chosen ECC should lie between the bit mismatch rate at the legitimate devices and that at the adversary.

We observe that the maximum BMR at the legitimate devices is 0.07 while the minimum BMR at the adversary is 0.48. Thus, an ECC whose error correction capability is above 0.07 while below 0.48 can guarantee that the legitimate devices establish a secret key that is not available to the adversary. In our experiment, we thus utilize RS(63,29), which can correct up to 17 symbols among a 63-symbol codeword (i.e., error correction capability of 0.27). To evaluate the key randomness, we run 8 randomness tests in the NIST test suite, which has 16 different statistical tests in total; the bit sequences generated in our work satisfy the input size recommendation of the chosen 8 tests only [29]. The result of each statistical test is typically in the form of a p-value, which must exceed 0.01 in order to pass a test. Table I shows the test results. We see the obtained p-values are all greater than 0.01.

TABLE II
MINIMUM, MAXIMUM, AVERAGE DRIVING DURATION AND DISTANCE TO ESTABLISH A KEY.

Traffic	L	Time (min)			Distance (km)		
		Min	Max	Avg	Min	Max	Avg
Light	128	1.20	3.77	1.52	0.95	3.24	1.49
	160	1.47	4.03	1.86	1.22	3.68	2.06
	192	1.73	4.30	2.19	1.76	4.26	2.44
Moderate	128	1.20	1.83	1.32	0.98	1.78	1.36
	160	1.47	2.10	1.63	1.12	1.99	1.63
	192	1.73	2.37	1.91	1.33	2.35	1.93
Heavy	128	1.22	2.80	1.53	0.86	1.69	1.34
	160	1.48	3.07	1.84	1.06	2.11	1.65
	192	1.75	3.33	2.15	1.34	2.47	1.95

F. Impacts of Traffic and Road

Different traffic situations (congestion levels) and road features (geometric design) may affect the driving behavior of a driver. We identify three distinguishable traffic patterns (i.e., heavy, moderate, and light), and choose three specific road segments, i.e., Curved, Straight, and a route with multiple turns at intersections (abbreviated Turns).

Under Same Road and Different Traffic: We find three traffic patterns during different time slots on regular weekdays for a typical commuting route. Our observation matches with the state transportation statistics. We choose 5-6 pm, 6-7 am, and 9-10 pm for representing heavy, moderate, and light traffic. The user then performs *DASK* when driving on the selected route. For each traffic situation, we perform 100 experiments.

Table II shows the required driving duration and distance. We observe that for each trip, the driving time (or distance) varies for establishing a key with a certain length, and a larger key size requires a longer driving distance or time for all traffic situations. Overall, moderate traffic takes less time regardless of the key size. This is because it has fewer redundant driving slots that do not contribute to the key establishment. Also, light traffic normally requires a longer driving distance due to the higher average speed and longer driving time.

Under Same Traffic and Different Road: Figure 12 depicts three selected road segments with different types. We perform *DASK* 100 times for each route with a moderate traffic pattern. Table III shows the obtained BGR on different road segments. The average BGR is in the increasing order of Turns, Curved,

Fig. 12. Three different road segments.

TABLE III
MINIMUM, MAXIMUM, AND AVERAGE BGRS.

Road Type	Bit Generation Rate (bit/sec)		
	Min	Max	Avg
Curved	0.6368	1.0	0.8835
Straight	0.5953	1.0	0.9848
Turns	0.4476	1.0	0.8000

and Straight. This is because, for the straight route, there are relatively fewer redundancy segments compared with the other two. For the route with turns, the user has to stop frequently, leading to the lowest BGR.

G. User Study

We recruited 10 volunteer drivers and asked each to perform *DASK* in their daily drive to establish a 128-bit key 100 times over a three-month period. Our study has been reviewed and approved by our institute IRB. Figure 13 shows the average driving duration and distance of 10 users. The required driving time and distance lie in the ranges of 1.07-1.27 min and 1.19-1.65 km. The corresponding standard deviations of the driving time and distance for different users are quite small (0.06 min and 0.18 km). Besides, we see that users 5, 6, and 9 present relatively short driving distances compared to other drivers while they require similar driving durations. This is because these three users drive more frequently during rush hours, with frequent velocity variations and a slower mean velocity. These results demonstrate convincingly that *DASK* is effective and robust in practice for generating secret keys.

V. LIMITATIONS

GPS dependency. Same with existing context-based key establishment schemes, which only apply to two devices equipped with the same corresponding sensors, *DASK* only works for GPS-enabled devices. However, GPS receiver is much more available and has been embedded into various devices, such as vehicles (e.g., Chevrolet [31]), smartphones, tablets, wearable devices, portable sensors (e.g., weather sensor PASCO PS-3209 [32]) and on-the-go medical alert devices (e.g., Mobile Guardian [33]).

The requirement of enough driving data. *DASK* sacrifices pairing time to remove user interaction during the key establishment. Currently, our implementation shows that it requires 1.52-minute driving to establish a 128-bit shared secret key, which is at the rate of generating 1.41 bits a second on average. The achieved bit generation rate in our scheme already has significant improvement over existing autonomous context-based pairing schemes (e.g., [2], [34]). For example, [2] acquires 56.6 bits within an hour (i.e., the bit generation rate ≈ 0.02 bit per sec) for device pairing. Also, [34] obtains

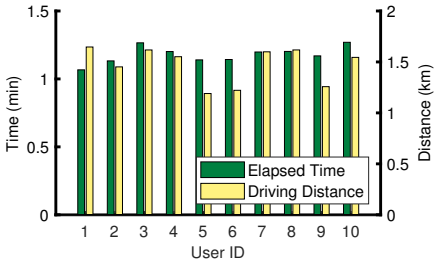

Fig. 13. Average driving time and distance for ten different drivers.

a bit generation rate of 368 bits per hour (≈ 0.10 bit per sec). Correspondingly, *DASK* is about 71 and 14 times faster than the above two techniques, respectively.

VI. RELATED WORK

Traditional secret key establishment: With fixed key management infrastructures, two devices may build a secure communication channel by taking advantage of traditional cryptography based methods (e.g., Diffie-Hellman key exchange). However, such methods are infeasible for the large numbers of mobile devices, with their limited resources. Also, many devices with dedicated capabilities such as Bluetooth Secure Simple Pairing [35] may require users to enter a passkey or a "yes/no" response to complete the device pairing. Such an interface is not always available, and human involvement may bring usability concerns as relying on it is cumbersome [36].

Context-based secret key establishment: There are emerging research efforts performing context-based secret key establishments, which utilize the randomness inherent in the observed environment and thus remove limitations of key management infrastructure in traditional cryptography. In particular, a pair of communicators equipped with the corresponding sensors observe the surrounding content, such as ambient sound [9], audio and luminosity [36], wireless channel state information [15], wireless signal strength [13], and wireless channel impulse response [14]. Two drawbacks hurdle the wide application of those approaches in practice. First, strict reciprocity is hard to achieve and non-reciprocal interference often brings a high error rate and slows the key establishment speed. Second, without extra device authentication solutions, those approaches suffer from MITM attacks [13], [14].

Some recent works propose countermeasures against MITM attacks when performing device pairing. Authors in [37] utilize speaker and microphone fingerprints due to manufacturing imperfection to defend against the MITM attack. However, this technique requires to pre-store the fingerprints of legitimate devices in order to authenticate them. Another work [2] uses the physical boundaries of a house to separate legitimate devices and adversaries and thus counteracts the MITM attack. Nevertheless, when the frequency of activities in the house is low, it will take a quite long time to build a secure connection between legitimate devices. *DASK*, unlike existing context-based device pairing schemes, neither needs to pre-share information between devices nor needs to create randomness deliberately. It extracts randomness in daily driving behaviors

and defends against MITM attacks by observing the coexistence of legitimate devices in a running vehicle.

Another study [38], based on a variable threshold-based quantization method, allows GPS devices co-located within a vehicle to derive the same key leveraging GPS data. On the contrary, *DASK* proposes a two-level interval based quantization technique and could achieve improved randomness for generated bits. Also, *DASK* is designed by considering stronger adversaries. Besides, we conduct more comprehensive experiments to evaluate *DASK* by investigating the impacts of traffic and road conditions as well as different drivers.

VII. Conclusion

We propose *DASK* for an autonomous and secure pairing of two devices using vehicle movement information from observed GPS readings. The insight is that multiple widely deployed navigation receivers presented in a running vehicle can capture the same rich randomness embedded in everyday driving behaviors, i.e., frequently and alternatively pressing the accelerator and brake pedals with varying force. Real-world experimental results with off-the-shelf GPS-enabled devices show that legitimate devices can successfully establish a 128-bit secret key with less than a 1.4 km drive on average under moderate traffic conditions. Meanwhile, an attacker following the target vehicle is unable to recover the established key.

Acknowledgement

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

References

[1] 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.

[2] J. Han, A. J. Chung, M. K. 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," in *IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 836–852.

[3] T. J. Pierson, T. Peters, R. Peterson, and D. Kotz, "Proximity detection with single-antenna IoT devices," in *ACM MobiCom*, 2019.

[4] Google Developers, "Automotive OS," https://developers.google.com/cars/design/automotive-os, 2022.

[5] S. Mirzadeh, H. Cruickshank, and R. Tafazolli, "Secure device pairing: A survey," *IEEE Communications Surveys Tutorials*, 2014.

[6] E. Uzun, K. Karvonen, and N. Asokan, "Usability analysis of secure pairing methods," in *Proc. of Int. Conf. on Financial Cryptography and Workshop on Usable Security*, 2007, pp. 307–324.

[7] J. Dunning, "Taming the blue beast: A survey of bluetooth based threats," *IEEE Security Privacy*, vol. 8, no. 2, pp. 20–27, 2010.

[8] 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.

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

[10] 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.

[11] W. Jin, M. Li, S. Murali, and L. Guo, "Harnessing the ambient radio frequency noise for wearable device pairing," in *ACM CCS*, 2020, pp. 1135–1148.

[12] A. Varshavsky, A. Scannell, A. LaMarca, and E. de Lara, "Amigo: Proximity-based authentication of mobile devices," in *UbiComp: Ubiquitous Computing*, 2007, pp. 253–270.

[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] 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.

[15] S. Fang, I. Markwood, and Y. Liu, "Wireless-assisted key establishment leveraging channel manipulation," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 263–275, 2021.

[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 *ACM WiSec*, 2020.

[17] M. Fomichev, J. Hesse, L. Almon, T. Lippert, J. Han, and M. Hollick, "Fastzip: Faster and more secure zero-interaction pairing," in *ACM MobiSys*, 2021, p. 440–452.

[18] 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.

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

[20] G. T. Amariucai, S. Barman, and Y. Guan, "An algebraic quality-time-advantage-based key establishment protocol," in *ACM WiSec*, 2018, pp. 144–153.

[21] National Coordination Office for Space-Based PNT, "GPS accuracy," https://www.gps.gov/systems/gps/performance/accuracy/, 2022.

[22] "Navstar GPS User Equipment," https://www.hsdl.org/?view\&did=22298, 1996.

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

[24] 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.

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

[26] 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.

[27] 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.

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

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

[30] Bushnell, "Velocity Speed Gun," https://www.bushnell.com/additional-products/speed-guns/velocity-speed-gun/BU-101911.html, 2022.

[31] Chevrolet, "In-vehicle technology," https://www.chevrolet.com/connectivity-and-technology/in-vehicle-technology, 2022.

[32] PASCO, "Wireless weather sensor with gps," https://www.pasco.com/products/sensors/environmental/ps-3209, 2021.

[33] Medical Guardian, "Mobile guardian," https://www.medicalguardian.com/products/mobile-guardian, 2021.

[34] M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Revisiting context-based authentication in IoT," in *Proc. of the 55th Annual Design Automation Conf. (DAC)*. ACM, 2018.

[35] 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.

[36] 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.

[37] D. Han, Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, "Proximity-proof: Secure and usable mobile two-factor authentication," in *ACM MobiCom*, 2018, pp. 401–415.

[38] E. Yang and S. Fang, "GPSKey: GPS-based secret key establishment for intra-vehicle environment," in *Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2022*, 2022.