WIP: Interrupt Attack on TEE-Protected Robotic Vehicles

Mulong Luo
Cornell University
ml2558@cornell.edu

G. Edward Suh
Cornell University
suh@ece.cornell.edu

***Abstract*—Effective coordination of sensor inputs requires correct timestamping of the sensor data for robotic vehicles. Though the existing trusted execution environment (TEE) can prevent direct changes to timestamp values from a clock or while stored in memory by an adversary, timestamp integrity can still be compromised by an interrupt between sensor and timestamp reads. We analytically and experimentally evaluate how timestamp integrity violations affect localization of robotic vehicles. The results indicate that the interrupt attack can cause significant errors in localization, which threatens vehicle safety, and need to be prevented with additional countermeasures.**

I. Introduction

For robotic vehicles (RV), sensor data are timestamped before used by control software. Timestamp integrity is defined as the consistency between timestamps attached to the sensor data and the groundtruth timestamps when the data are captured by the sensor in the physical world. If the differences between these values are within a bound, then timestamp integrity is maintained, otherwise it is violated. Timestamp is used in Kalman filter-based [18] and deep learning-based [16], [15] multi-sensor fusion. It is important in robotic vehicles because an incorrect timestamp can lead to wrong perceptions and actuation in the physical world, potentially causing irreversible physical damages.

Timestamps are data stored in memory, and a compromised application or an operating system (OS) may modify them in memory. Besides, the source clock maybe tampered. What is more, protecting the integrity of memory and clock is not enough for timestamp integrity. Timestamp integrity has unique properties that are different from the traditional data integrity that only require trusted source and untampered data. A timestamp must be read from a clock at the time when the corresponding sensor input is taken.

Some existing trusted execution environments provide protection of data integrity against untrusted or compromised OS, besides, trusted clock [6] already exists in the latest commercial TEE such as SGX2 [1]. However, we showed in this paper, that existing TEEs cannot protect timestamp integrity. We demonstrate *interrupt attack* that injects a delay between reading a sensor inputs and reading a timestamp clock, which can compromise the timestamp integrity and potentially cause significant impacts on robotic vehicles' perception and behaviors. RV software typically reads a sensor input and a timestamp clock sequentially. Ideally, these two operations should be performed atomically. However, that is not guaranteed by existing hardware. In the interrupt attack, a compromised OS injects an interrupt between the sensor read and the timestamp clock read, so that there is a long delay between them. As a result, the timestamp attached to the sensor data becomes significantly different from the ground truth timestamp.

In this paper, we study the impact of interrupt attack on TEE-protected robotic vehicles. The main contributions of this paper are as follows:

1) We analytically derived the relationship between timestamp errors and the physical-world localization errors;
2) We designed and implemented a TEE-based sensor timestamp module using commercial-off-the-shelf hardware and demonstrate the effect of the interrupt attack on the timestamp integrity;
3) We demonstrated the interrupt attack on the ROS platform, and provide quantitative studies on its impact on localization errors.

Our preliminary results demonstrate proof-of-concept interrupt attack on ROS, and we plan to show real-world impact on commercial-grade autonomous driving platform like Apollo [2] as future work. To protect interrupt attack, TEE requires disabling interrupt from the OS. However, not all existing commercial secure enclaves provide that capability. We plan to evaluate some recent research prototype [5] which is able to disable interrupt for secure enclave on commercial RV platform.

II. Threat Model

We target the robotic vehicle that runs a sensing application and needs timestamps for processing different sensor data. The goal of the attacker is to stealthily change the control decision of a RV stealthily by compromising timestamp values in software. We assume that the attacker is capable of exploiting software vulnerabilities on the RV and taking control of an OS. On the other hand, we assume that the adversary has no physical access to the system and the sensor inputs from the physical world is not affected by the adversary, i.e., there is no physical spoofing.

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.2022.23001
www.ndss-symposium.org

To protect against traditional software attacks, we assume that the sensor processing software that performs timestamping runs inside a TEE such as Intel SGX. We also assume that the TEE's clock is not tampered with and has secure IO mechanisms to read sensors and control actuators. In that sense, the adversary cannot directly change senor inputs or timestamp values, and cannot compromise the integrity of the control software. On the other hand, we assume that a compromised OS can still interrupt RV software during its execution inside a TEE (enclave). Note that today's TEEs rely on an OS for scheduling processes and performing complex IO in order to reduce the size of the trusted computing base (TCB) and allow efficient sharing of a platform between secure and normal applications.

III. Timestamp Integrity in RV

Let us assume that the vehicle locations are $(t_v^1, \mathbf{x}_v^1), ..., (t_v^n, \mathbf{x}_v^n)$, where $\mathbf{x}_v^i$ is the location of the vehicle v corresponding to the static frame at time t_v^i, and that the position of a point p correspond to the vehicle v in the pointcloud returned by the LiDAR is $\mathbf{x}_{p,v}$, at time t_p. As the LiDAR sampling time t_p is usually not synchronized with the odometry sampling time, the location of the vehicle at time t_p needs to be interpolated. Using linear interpolation, the vehicle location $\mathbf{x}(t_p)$ corresponding to the static frame can be represented as:

$$\mathbf{x}(t_p) = \frac{\mathbf{x}_v^j(t_v^{j+1} - t_p) + \mathbf{x}_v^{j+1}(t_p - t_v^j)}{t_v^{j+1} - t_v^j} \tag{1}$$

Then, the location of the point p in reference to the static frame $\mathbf{x}_p$ can be represented as

$$\mathbf{x}_p = \mathbf{x}(t_p) + \mathbf{x}_{p,v} = \frac{\mathbf{x}_v^j(t_v^{j+1} - t_p) + \mathbf{x}_v^{j+1}(t_p - t_v^j)}{t_v^{j+1} - t_v^j} + \mathbf{x}_{p,v} \tag{2}$$

where $t_v^j \leq t_p \leq t_v^{j+1}$.

Thus, a Δt_p change in t_p will result in $\Delta \mathbf{x}_p$ in location represented by the following:

$$\Delta \mathbf{x}_p = \frac{\mathbf{x}_v^{j+1} - \mathbf{x}_v^j}{t_v^{j+1} - t_v^j} \Delta t_p = \mathbf{v} \Delta t_p \tag{3}$$

where $\mathbf{v}$ is the velocity of the vehicle at time t_v^j. Equation (3) indicates that the localization error of an obstacle increases with the speed of the vehicle as well as the timestamp error.

IV. Interrupt Attack on Timestamp Integrity

The interrupt attack can compromise timestamp integrity even in the presence of a hardware-protected trusted execution environment (TEE) and a trusted clock. Today's TEEs provide strong protection for confidentiality and integrity from a compromised OS. With a TEE, an OS cannot modify the timestamp in memory directly. In addition, with a trusted clock source, the attacker cannot affect the timestamp value that the protected software inside a TEE reads from the clock. Figure 1 shows an architecture that uses a TEE (secure enclave) and a trusted clock to protect timestamp integrity. Raw data arrive from a sensor via a trusted IO [14] interface. The timestamp module reads a trusted clock to obtain a timestamp for the sensor data, and the timestamped data are sent to the control software via a secure communication channel.

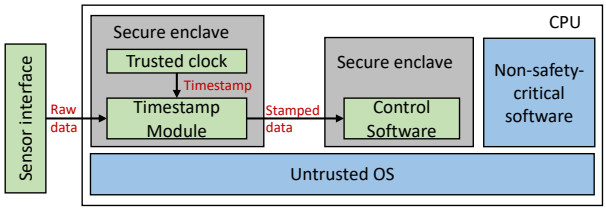

Fig. 1. An architecture that uses TEE (secure enclaves) to protect the data integrity of the timestamp, the code shown in Listing 1 runs in the left secure enclave.

Listing 1 shows the conceptual code of the timestamp module that runs inside a secure enclave. The function `OnSensor()` creates object `StampedData sd`, reads the raw sensor data `sd.d` via trusted IO using `getSensorData()`, then it gets the timestamp `sd.t` and returns the object `sd`. Since the timestamp and the sensor data are from a trusted clock and a trusted IO respectively, the values are trusted. Besides, the untrusted OS cannot modify object `sd` or the timestamp `sd.t` since the code is executed inside a secure enclave.

Listing 1. Conceptual code for timestamping sensor data.

```
struct {
  int64 d; //data measurement
  int64 t;  //timestamp of the data
} StampedData;

StampedData OnSensor() {
  StampedData sd; //create object
  sd.d = getSensorData();
  // read sensor from trusted IO
  sd.t = getTimestamp();
  // get the timestamp
  return sd;
  // function returns
}
```

However, an interrupt attack can still violate the timestamp integrity even with the secure enclave, the trusted clock, and the trusted IO. The interrupt attack is performed by a compromised OS that injects interrupts between `getSensorData()` and `getTimestamp()`, such that there is a long delay between them. As a result, the timestamp `sd.t` being attached to the sensor data `sd.d` is no longer representative of when the data is actually sampled.

Because an OS controls the scheduler, a compromised OS can raise interrupts inside an enclave. There are multiple instructions inside `getSensorData()` and `getTimestamp()`. The attacker can inject multiple interrupts in to compromise the timestamp integrity.

Since sensor timestamping is just a small portion of the entire control program, a successful interrupt attack have to interrupt only the execution of the enclave but not other parts. (Otherwise, the entire control program will execute very slow, making the attack easy to be discovered.) We can achieve the interrupt attack on timestamp integrity using SGX-Step [24], which injects multiple interrupts in secure enclave with a relatively short duration. Figure 2 shows the result. By decreasing the SGX-Step timer interval `SGX_STEP_TIMER_INTERVAL`

(which controls how often the program is interrupted), more instructions are interrupted and the timestamp error increases. When the parameter `SGX_STEP_TIMER_INTERVAL` is 127, the timestamp error is hundreds of milliseconds and sometimes greater than 1 second. As we will see in Section V, even tens of milliseconds can cause significant localization errors to a robot.

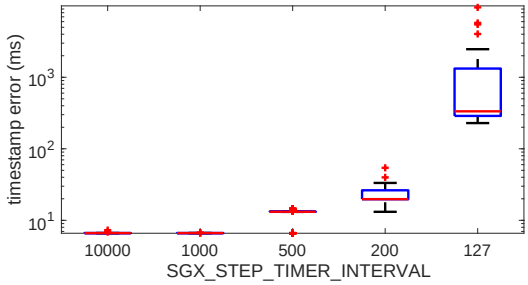

Fig. 2. Timestamp error vs the SGX step timer interval.

V. Evaluation of Interrupt Attack on RV

In this section, we demonstrate the impact of the interrupt attack with a varying length (latency) on the localization of a vehicle and obstacles using ROS [3]. Many commercial autonomous driving platforms, such as Baidu Apollo shares similar low-level architecture. We perform our experiments on an Intel STK2MV64CC computer with a SGX-capable dual-core processor (Core m5-6Y57), running Ubuntu 16.04. We simulate a Jackal robot [4] in the Gazebo simulator, in which Jackal's software is controlled by ROS. The maximum speed of the robot is set to be 0.5 m/s. The LiDAR periodically scans every 100 ms, and we tested interrupt attacks with varying latency from 0 ms to 100 ms. The interrupt attack did not affect the frequency of the LiDAR scans[1]. Figure 3 shows the map used for the experiments.

We implement an TEE-protected timestamp module in ROS. To mimic the interrupt attack caused by an adversarial OS while controlling timestamp error more precisely, we force the timestamp module to sleep for a specific duration before the timestamp is returned. Effectively, the sleep duration can represent the interrupt latency an attacker injects. we use ROS time for timestamps[2].

Impact on Obstacle Localization: In this experiment, the vehicle gets its own location from the odometry. In order to navigate through the obstacles, the vehicle has to localize the obstacles on a given map. The obstacle localization is based on the LiDAR. Since the obstacles (walls on map) are distributed continuously, we do not directly measure the localization error for each individual obstacle. Instead, we compare the actual path the vehicle is taking to reflect the obstacle localization errors. In a straight hallway (from A to B in Figure 3) surrounded by two parallel walls, without any obstacle localization error, the vehicle should move in a straight line. However, obstacle localization errors caused by the interrupt attack sometimes change the perceived wall locations to the center of the straight path, which causes the vehicle to deviate from the straight path to avoid collision with the perceived wall.

[1]For each LiDAR scan, ROS invokes a new thread to process and timestamp the input. For each thread, even though the sum of the interrupt latency and the execution time might be greater than the scanning period, the next scan does not have to wait until the processing of the first scan is finished.

[2]We assume ROS time is not tampered with by the attacker.

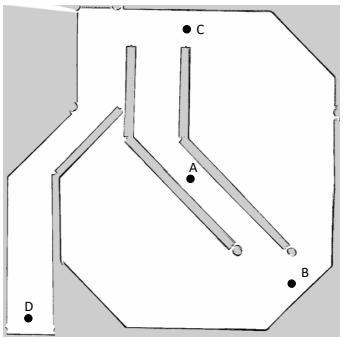

Fig. 3. A Jackal race track map used for evaluating the impact of the interrupt attack. The white space represents flat ground the vehicle can move on while the grade space represents the walls/obstacles the vehicle needs to avoid. The coordinates of A, B are (0,0), (-7,7), (0,8) and (-9, -9), respectively.

We consider the case when the vehicle is initially at A facing south and the destination is B on Figure 3. Figure 4 and Figure 5 show the straight paths and actual paths of the vehicle when there is no interrupt attack (0 ms interrupts) and when there is an interrupt attack with 100 ms interrupts, respectively. The maximum deviation between the straight path and the actual path is 0.171 m and 0.647 m, respectively. A longer interrupt leads to a larger error in obstacle localization, which drives the actual path farther from the straight line.

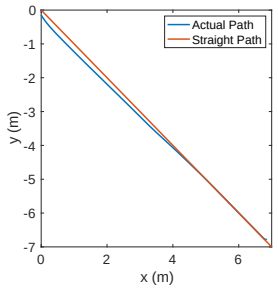

Fig. 4. The vehicle path with no interrupt attack. The maximum deviation from the straight line is 0.171 m

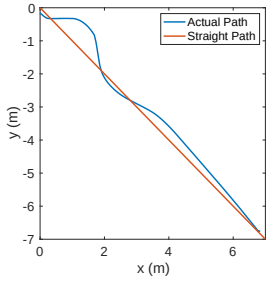

Fig. 5. The vehicle path with 100 ms interrupts. The maximum deviation from the straight line is 0.647 m

Figure 6 shows the average and maximum deviations of the actual path from the straight path. The figure shows that the deviation generally increases with the interrupt latency (length). However, we see a drastic increase when the latency is 100 ms. As shown from Figure 5, the actual path of the vehicle becomes unstable, oscillating around the straight path.

Impact on Ego Localization: We use AMCL for ego localization, which uses both odometry and LiDAR inputs to localize the vehicle's location. The vehicle initially is at point A and moves to B, C, and D sequentially. Because ego localization interacts with path planning in a close loop, an error in ego localization leads to a different planned path, which in turn further affects the localization error. To exclude the effect of path planning and quantifying only the impact of interrupt latency on ego localization, we use trace-based experiments

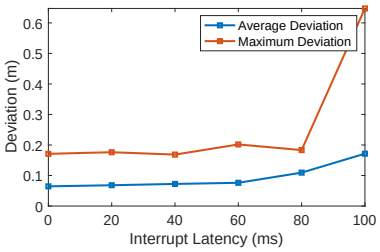

Fig. 6. Average obstacle localization error vs interrupt latency.

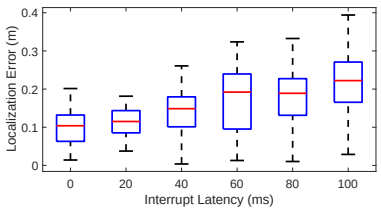

Fig. 7. Ego localization error distribution vs interrupt latency.

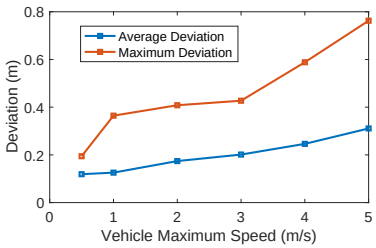

Fig. 8. The average deviation vs the speed of the vehicle when the interrupt latency is 90 ms.

where we first record raw (i.e., without timestamp) LiDAR and odometry data traces without an interrupt attack, and replay the traces to have the localization module perform the localization task. Because the experiments use recorded traces, there is no path planner running during localization and the odometry trace does not change in response to the localization errors. Thus, the error between the ground truth location and the estimated location of the vehicle is purely from the localization error at each point and not affected by the path planner.

We show the relationship between the localization error distributions along the path and the interrupt latency in Figure 7. The average and maximum localization errors along the path increases with the interrupt latency. The maximum localization error almost doubles to around 0.4 m when the interrupt latency is 100 ms compared to around 0.2 m when the latency is 0 ms.

Figure 9 and Figure 10 also show examples of how the ego localization errors, when coupled with the path planner, can be dramatically magnified and significantly change the vehicle's path. The vehicle starts at (0,0) towards the destination at (6, -7). When the interrupt latency is 30 ms, the vehicle can reach the destination with only small errors in localization. However, when the interrupt latency increases to 60 ms, not only the localization error is obvious, but the vehicle fails to reach the destination because of the large localization error.

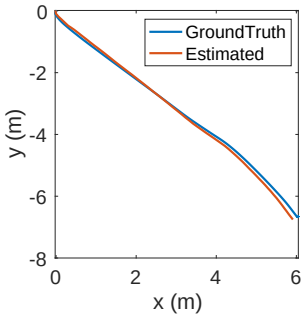

Fig. 9. Ground truth and estimated location of the ego vehicle when interrupt latency is 30 ms. The average error is 0.095 m.

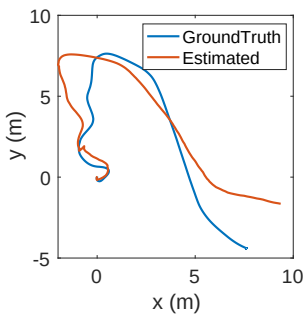

Fig. 10. Ground truth and estimated location of the ego vehicle when interrupt latency is 60 ms. The average error is 1.45 m.

Impact of Vehicle Speed: As Equation (3) shows, theoretically, the localization error is proportional to the velocity $\mathbf{v}$ of a vehicle. We show the impact of the vehicle's speed on the impact of the interrupt attack. Figure 11 and Figure 12 show the path of a vehicle when the speed of the vehicle is 0.5 m/s and 5 m/s, respectively, with the interrupt attack

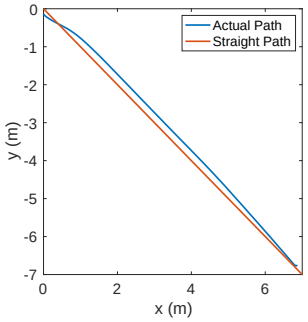

Fig. 11. The actual path and the straight path of a vehicle when the maximum speed is 0.5 m/s.

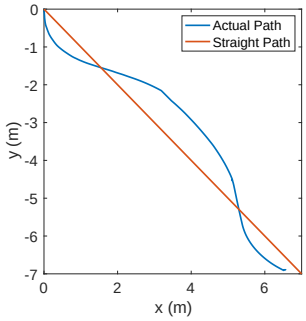

Fig. 12. The actual path and the straight path of a vehicle when the maximum speed is 5 m/s.

with is 90 ms interrupts. Figure 8 shows the average and the maximum deviation of the vehicle when the vehicle has a different maximum speed. The general trend shows that both the average deviation and the maximum deviation increase with the speed. A vehicle often moves at a higher speed in the real world (e.g., 30 m/s on a highway), and the deviation can be as much as several meters in such cases. Considering the lane width is only about 4 m, such a large deviation can be a serious concern for the safety of a vehicle.

VI. Related Work

The importance of accurate timestamps has been discussed in the context of wearable sensors for healthcare [19]. For more accurate timestamps, previous work also reads timestamps twice at the beginning and the end of the execution [17]. This approach was applied to mobile health monitoring [20] and blockchain [25]. While the previous studies discussed timestamp integrity in sensor inputs, this paper represents the first to discuss and evaluate the timestamp integrity problem in the context of RVs.

We perform the interrupt attack and evaluate the impact on RVs. There are other time-related attacks that may affect RV safety. In the Butterfly attack [12], the attacker leverages variations in computation time to destabilize UAVs. In Out of Control [8], the attacker creates an artificial delay by running computational intensive applications on the same platform to divert robot vehicles. In the time delay attack [11], the attacker threatens smart grid stability by delaying the communication packets that transport the state information among nodes. These attacks show the importance of time in RV safety, but are orthogonal to the timestamp integrity.

Secure hardware techniques include secure enclave, trusted clock, trusted IO, and availability protection. Many of them are necessary for timestamp integrity protection. Intel SGX provides a trusted clock with the precision of seconds for the enclave. However, a "secure" version of a high-precision clock was removed in the latest version due to its insecurity. Without a trusted hardware clock, people are using a software clock inside an enclave [6], [22] as a trusted high-precision clock. We need the trusted IO capability that directly connects a secure enclave to a sensor input without allowing a compromised OS to modify it. Trusted IO for SGX is proposed in [14]. In [26], the authors provide integrity of IO configurations and firmware leveraging system management mode. ARM TrustZone, on the other hand, has native support for secure IO [10], which is used in sensors for connected vehicles [9].

The interrupt attack shows how the timestamp integrity may be compromised even when protected by a TEE. The previous studies also identified other attacks on the TEE. For example, Foreshadow [23] and SGXSpectre [7] compromise confidentiality using a covert channel. PlunderVolt [13] and CLKScrew [21] show that the integrity of a TEE can be broken by exploiting a hardware vulnerability. These vulnerabilities are mitigated by firmware updates from processor vendors. Interrupt attack, on the other hand, is hard to be directly fixed by a simple firmware update. Implementing a novel interrupt policy that prevents interrupt attack while not overprivileging the enclave program might require hardware change.

VII. Conclusion and Future Work

In this paper, we investigated the interrupt attack on the timestamp integrity of TEE-protected RV. The interrupt attack can break the consistency between the timestamp and sensor data without overwriting a timestamp value in memory, effectively violating timestamp integrity. We experimentally studied the impact of interrupt latency on the ego and obstacle localization errors, which pose a threat to vehicle safety. This study suggests that we need additional protection for timestamp integrity even when RV software is protected by a TEE. We plan to implement TEE-protected timestamping in the state-of-the-art autonomous driving platform like Apollo and demonstrate the attack it and evaluate the consequences. We also plan to evaluate some recent research TEEs which have instructions for disabling interrupt for RV timestamp integrity protection.

References

[1] [Online]. Available: https://www.intel.com/sgx

[2] [Online]. Available: https://apollo.auto

[3] [Online]. Available: https://www.ros.org

[4] [Online]. Available: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

[5] F. Alder, J. Van Bulck, F. Piessens, and J. T. Mühlberg, "Aion: Enabling open systems through strong availability guarantees for enclaves," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1357–1372.

[6] F. M. Anwar, L. Garcia, X. Han, and M. Srivastava, "Securing time in untrusted operating systems with timeseal," in *2019 IEEE RTSS*. IEEE, 2019, pp. 80–92.

[7] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *2019 IEEE EuroS&P*. IEEE, 2019, pp. 142–157.

[8] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Out of control: stealthy attacks against robotic vehicles protected by control-based techniques," in *ACSAC*, 2019, pp. 660–672.

[9] S. Hu, Q. A. Chen, J. Joung, C. Carlak, Y. Feng, Z. M. Mao, and H. X. Liu, "Cvshield: Guarding sensor data in connected vehicle with trusted execution environment," in *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*, 2020, pp. 1–4.

[10] M. Lentz, R. Sen, P. Druschel, and B. Bhattacharjee, "Secloak: Arm trustzone-based mobile peripheral control," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, 2018, pp. 1–13.

[11] X. Lou, C. Tran, R. Tan, D. K. Yau, and Z. T. Kalbarczyk, "Assessing and mitigating impact of time delay attack: a case study for power grid frequency control," in *ACM/IEEE ICCPS*, 2019, pp. 207–216.

[12] R. Mahfouzi, A. Aminifar, S. Samii, M. Payer, P. Eles, and Z. Peng, "Butterfly attack: Adversarial manipulation of temporal properties of cyber-physical systems," in *RTSS*. IEEE, 2019, pp. 93–106.

[13] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against intel sgx," in *IEEE S&P'*, 2020.

[14] T. Peters, R. Lal, S. Varadarajan, P. Pappachan, and D. Kotz, "Bastion-sgx: Bluetooth and architectural support for trusted i/o on sgx," in *HASP*, 2018, pp. 1–9.

[15] S. S. Sandha, L. Garcia, B. Balaji, F. M. Anwar, and M. Srivastava, "Sim2real transfer for deep reinforcement learning with stochastic state transition delays."

[16] S. S. Sandha, J. Noor, F. M. Anwar, and M. Srivastava, "Time awareness in deep learning-based multimodal fusion across smartphone platforms," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 149–156.

[17] S. Saroiu and A. Wolman, "I am a sensor, and i approve this message," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, 2010, pp. 37–42.

[18] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under gps spoofing," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 931–948.

[19] M. Siddiqi, V. Sivaraman, and S. Jha, "Timestamp integrity in wearable healthcare devices," in *2016 IEEE ANTS*, pp. 1–6.

[20] J. M. Sorber, M. Shin, R. Peterson, and D. Kotz, "Plug-n-trust: practical trusted sensing for mhealth," in *ACM/IEEE MobiSys*, 2012, pp. 309–322.

[21] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in *USENIX Security*, 2017, pp. 1057–1074.

[22] B. Trach, R. Faqeh, O. Oleksenko, W. Ozga, P. Bhatotia, and C. Fetzer, "T-lease: a trusted lease primitive for distributed systems," in *Proceedings of the 11th ACM SoCC*, 2020, pp. 387–400.

[23] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *USENIX Security*, August 2018.

[24] J. Van Bulck, F. Piessens, and R. Strackx, "SGX-Step: A practical attack framework for precise enclave execution control," in *SysTEX*. ACM, Oct. 2017, pp. 4:1–4:6.

[25] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM CCS*, 2016, pp. 270–282.

[26] F. Zhang, "Iocheck: A framework to enhance the security of i/o devices at runtime," in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*. IEEE, 2013, pp. 1–4.