Suggested Topics (almost all would require lab presence)

1) Examine the use of TRNGs and PUFs (hardware fingerprints) in security protocols

- Generate TRNG & PUFs, e.g., from uninitialized SRAMs
- Experiment with TRUE RNG hardware (TRNG chip, INFNOISE)
- Develop a fingerprint-related protocol and prepare a demo
- Examine their use in authentication protocols, e.g., generating session keys

2) RFID-related security protocols

- Learn about Secure Electronic Transactions (SET)
- Experiment with SET protocols using TLS & possibly Zymbit zymkey crypto IC

3) RFID smart cards: technology & attacks

- Examine tag architecture, security
- Discover innovative security protocols that relate to cheap RFID cards, such as Mifare Classic
- Build custom security protocols, use of multiple tags
- Integrate RFID tech to perform security challenges, enable vehicle diagnostics (OBD), or patch running code on embedded systems
- Use Proxmark3 to reverse engineer & attack, e.g., sniff, replay or clone different smartcards

4) Examine and evaluate key loggers

- Examine the design of different hacking mechanisms
- Experiment with user-level apps and Linux kernel rootkit modules
- Examine related hacking software (e.g., metasploit, openvas, etc) - experiment with VMs
- Use rubber ducky dongle and other similar hardware devices for hacking from Hak5

5) Hardware Security Modules: Zymbit zymkey and ATECCx08 crypto devices

- Integrate Zymbit zymkey with CAN bus requests/replies or with TLS
- Integrate ATECC608a with CAN bus or TLS
- Explore the Microchip cryptoauth world: provisioning & protocols (experiment with ATECC508 on Arduino, RPI or Dragonboard 410c)

6) ARM Trustzone

- Study ARM Trustzone sw development (secure, rich world), Cheri/ARM Morello architecture
- Examine ARM Trustzone design technologies and abstractions (e.g., Trustonic)
- Experiment with Microchip ATSAML11 using Microchip Studio, MPLAB
- Examine ARM Trustzone implementation on USB Armory (secure boot, etc)

7) Apply wireless hacking tools with Hack RF

- Use GNUradio Companion and other tools to capture wireless communication patterns
- Consider Bluetooth (using BT devices), IR (using IR Anavi), TV signals etc

8) Add data privacy & security to an E-health application

- Use open source heartypatch device (a cardiac sensor)
- Develop secure biosensor driver/application that runs on 2-core ESP32
- Development in FreeRTOS of authentication & encryption and anonymity protocols

9) Experiment with covert channels

- Discover interesting scenarios and implementations of covert channels
- Experiment with software channels, e.g., networking, audio
- Examine hardware channels, e.g., optical (optisense device and tomu/arduino led)

10) Secure probing on (I2C, SPI) and UART peripherals

- Examine how to design secure sensors (over I2C, SPI or UART)
- Examine the status of related patents etc
- Develop a prototype related to secure access to sensors
- Demonstrate using a prototype with probes (Aardvark, Beagle) or OBD dev kit

11) Secure network logs and actions in Linux

- Examine syslog and Secure Event Correlator for correlating network logs and honeypot data
- Develop a framework with dynamic security rules (regular expressions) and actions on an embedded platform, e.g., Odroid XU4
- Examine real-time visualization, e.g., heatmaps

12) CAN Bus: security attacks and solutions in automotive networks

- Examine Security & Safety Standards in Automotive Systems
- Design attacks and examine/develop detection/protection (IDPS) mechanisms over CAN bus
- Develop a secure travel datalogger from malicious third-party exploits
- Experiment on existing platform that connects ECUs to ECUSIM2000 (engine simulator)

13) Dangerous driving notification system

- Develop detection & notification system based on a driver profile characterizing driving behavior
- Voice interface solutions with CAN Bus (local & internet-based text-to-speech & speech-to-text)
- Connect to mobile telematics applications in real-time, e.g., auto insurance and driver verification
- Experiment on existing platform that connects ECUs to ECUSIM2000 (engine simulator)

14) Software vulnerabilities - stack & heap overflows

- Examine stack & heap overflow
- Study stack smashing attacks
- Study heap attacks on physical memory using pagemap analysis
- Develop a demo attack on a device driver that stores sensitive data sent from a biosensor in a statically or dynamically memory allocated array

15) Network firewalls

- Examine network firewall and network packet capture technology, including kernel support
- Examine related bandwidth management techniques
- Examine methods, filters & languages for specifying rules, e.g., iptables, netfilter ufw, BPF, etc
- Examine distributed firewalls and related open issues

16) Experiment with software stacks on small systems, e.g., Arduino & SBCs (Single Board Computers)

- Examine, evaluate, and extend lightweight IPC
- Protothreads and Linux-based user-level threads
- Lightweight Network Stacks
- Small Database πSystems

17) Improving sampling rate of Biosensor running FreeRTOS using encoding and IPC

- Use custom IPC protocols (task management, IPC, sockets) to improve the sampling rate
- Exploit encoding and data compression with/without data loss
- Evaluate tradeoffs in performance, reliability, and accuracy

18) Design methodology and tools for automated detection/protection of/from hardware trojans

- Examine methodology & tools for detecting sensitive data and locate possible hardware trojans
- General Binary Decision Diagram examples,
- TRNG and PUFF example
- FPGA NoC Firewall prototype

19) Experiment with virtual channels on a packet router in SystemC

- Examine the use of virtual channels on packet switches
- Extend an existing 2 x 2 SystemC switch to use virtual channels
- Develop a network-on-chip using these switches
- Simulate and derive its latency, performance, and fairness characteristics
- Consider synthesis using Vivado HLS tools

20) Develop a new Linux “mixed criticality” CPU scheduling policy

- Study the Linux kernel scheduler and related data structures
- Use a prototype environment on Zedboard to examine a “mixed criticality” Linux scheduler
- Develop examples and evaluate its performance using internal debugging and external probes
- Try to extend its policies

21) Develop IoT applications using Python

- Prototyping sensor- and actuator-based applications on Raspberry Pi, 96boards Dragonboard 410C or Hikey 620, and/or Microchip Boards
- Exploiting RPC for sending expensive functions to a computing cluster
- Consider cluster-based management: load balancing, etc by using high-level tools
- Apply a simulated annealing environment to guide a Kubernetes scheduler

22) Reverse Engineering

- Methods and tools
- Prototype different profiling, tracing, debugging, probing tools
- Measure entropy for identifying keys, e.g., using binwalk, bindiff on images
- Experiment with Ghidra

23) Real Time and Vector Clocks

- One way Delay
- Clock Synchronization
- Vector Clocks

Reverse Engineering etc

existing privacy standards and regulations as well as ongoing efforts in the automotive domain and survey the landscape of automotive data-privacy attacks which can be classified into three categories: driver fingerprinting, location inferencing and driving-behavior analysis. After designing a threat model for third-party vehicular data sharing, we will then investigate an example location-inference attack using less obvious sensors since the thus-collected data can leak users' location and traveled routes without explicitly using the GPS readings. We will also analyze the accuracy of inferring the user's location from vehicle data and show the impact of this on the car owner's privacy. Furthermore, we will show how this privacy vulnerability can be mitigated by designing a privacy-preserving data-collection architecture and schemes using differential privacy for vehicles.

Detecting Misplaced RFID Tags on Static Shelved Items

This project aims to detect misplaced tags on stationary shelved items so that we can minimize out-of-stock and reduce warehousing and labor costs. Our idea is to make the measured phase coincide with the theoretical phase and identify the phase shifts caused by misplacement of tags. However, this is still challenging due to position ambiguity, phase wrapping, device diversity, and phase ambiguity. To overcome these challenges, we propose an effective way of detecting misplaced tags, called *FINDS<\i>, which requires neither antenna movement nor external disturbances.*

CPS: Breakthrough: Secure Interactions with Internet of Things

The objective of this research is to (1) gain insights into the challenges of securing interactions in Internet of Things (IoT) deployments, (2) develop a practical framework that mitigates security and privacy threats to IoT interactions, and (3) validate the proposed framework in a medium-scale IoT testbed and through user studies. The emerging IoT computing paradigm promises novel applications in almost all sectors by enabling interactions between users, sensors, and actuators. These interactions can take the form of device-to-device (e.g., Bluetooth Low Energy (BLE)) or human-to-device (e.g., voice control). By exploiting vulnerabilities in these interaction surfaces, an adversary can gain unauthorized access to the IoT, which enables tracking, profiling and posing harm to the user. With the thousands of diverse IoT manufacturers, developers, and devices, it is very challenging, if not impossible, to ensure all devices are properly secured at production and kept up-to-date after production. IoT users and administrators have to place their trust in a set of devices, with the least secure device breaking the security chain.

Collision Avoidance

Recent incidents have shown that both human control and autonomous control of the vehicles cannot be entirely trusted. To provide a safer driving experience, we propose a system utilizing V2X communication that is able to resolve conflicting control commands between the human driver and the autonomous system by introducing Edge as a tie-breaking entity. The system features data integrity cross-validation by utilizing both cyber-physical and wireless measure characteristics and it is designed under the assumption that any received or perceived data can be manipulated.

Sensor Integrity Verification

Vehicle Anomaly Diagnostic System
To provide safer and more comfortable driving experience, more and more electronic

components are added to cars. However, their integration also increases the system complexity and introduces new attack surfaces, thus making it harder to identify the source/cause of anomalous vehicle behavior. To alleviate this difficulty, we have been developing a vehicle anomaly diagnostic system, called CarDog, to trace and hunt/narrow down the source(s) of anomaly. Specifically, CarDog is designed to identify the source(s) of anomaly related to vehicle maneuvers, including input-to-response consistency and powertrain operation. It constructs hierarchical detection sets (DSs) that overlap with each other. By capturing the cyber-physical characteristics in each DS, CarDog is able to cross-validate the vehicle data with multiple combinations and identify the faulty data/components even if no data can be entirely trusted.

Context-Aware Sensor Integrity Verification
While more and more driver assistance function added to modern vehicles, they also introduce new attack surfaces to the vehicle. The goal of this project is to develop a system that is able to detect and verify context information, such as road inclination, for verifying vehicle IMU readings and location information to prevent the control system or service provider from producing an incorrect decision. The system captures the context information by monitoring the behavioral change between different vehicle components. The system then compares the captured change to the road/map information to verify the data integrity or identify the occurrence of an anomaly.

Automotive Data Translation

Modern connected and autonomous vehicles (CAVs) are equipped with an increasing number of Electronic Control Units (ECUs), with many producing large amounts of data. This data is exchanged between ECUs via an in-vehicle network, with the Controller Area Network (CAN) bus being the de facto standard in current vehicles. Furthermore, CAVs do not only have physical interfaces, but also increased data connectivity to the Internet via their Telematic Control Units (TCUs), which enables them to be accessed remotely like mobile phones. In addition, since data transmitted on the CAN bus is not encrypted, it is possible to tap into and read/write data from/to the CAN bus. This naturally raises concerns about automotive cybersecurity. One commonality among most recent reported vehicular security attacks is that they ultimately require write access to the CAN bus. Nevertheless, malicious packets injected onto the CAN bus can only cause targeted and intentional changes in vehicle behavior if the attacker has knowledge of the CAN message format. However, since this format is proprietary to OEMs and can differ even among different models of a single make of vehicle, one must manually reverse engineer the CAN message format of each vehicle they target — a time-consuming and tedious process that does not scale. In this project, we develop a tool which can translate most CAN messages with minimal effort.

Automotive Data Privacy: Attacks and Defenses

Vehicular data-collection platforms as part of OEM's connected telematic services are on the rise. OEMs use these platforms to derive various telematics parameters from the in-vehicle network for connected services. OEMs also allow sharing of the collected data with third-party service providers upon request. We propose to investigate the broad field of automotive data privacy since it has been a rapidly rising concern. The European Union has established a privacy standard called General Data Protection Regulation (GDPR) in May 2018. Furthermore, the Facebook-Cambridge Analytica data incident made headlines in March 2018. OEMs' data collection from vehicles is increasingly popular and may offer OEMs new businesses but it comes with the risk of privacy leakages. Vehicular sensor data shared with third-parties can lead to misuse of the requested data for other purposes than stated/intended.

In this project, we first provide an overview of existing privacy standards and regulations as well as ongoing efforts in the automotive domain and survey the landscape of automotive data-privacy attacks which can be classified into three categories: driver fingerprinting, location inferencing and driving-behavior analysis. After designing a threat model for third-party vehicular data sharing, we will then investigate an example location-inference attack using less obvious sensors since the thus-collected data can leak users' location and traveled routes without explicitly using the GPS readings. We will also analyze the accuracy of inferring the user's location from vehicle data and show the impact of this on the car owner's privacy. Furthermore, we will show how this privacy vulnerability can be mitigated by designing a privacy-preserving data-collection architecture and schemes using differential privacy for vehicles.

Defensive Routing: a Preventive Layout-Level Defense Against Untrusted Foundries

Since the inception of the integrated circuit (IC), the size of the transistors used to construct them continually shrink. While this advancement significantly improves computing capability, the associated massive complexity forces IC designers to outsource fabrication. Outsourcing presents a security threat: comprehensive post-fabrication inspection is infeasible given the size of modern ICs, thus it is nearly impossible to know if the foundry has altered your design during fabrication (i.e., inserted a hardware Trojan). Defending against a foundry-side adversary is challenging because---with as little as two gates---hardware Trojans can completely undermine software security. Prior work attempts to both detect and prevent such foundry-side attacks, but all existing defenses are ineffective against the most advanced hardware Trojans.
We present Defensive Routing (DR), a preventive layout-level defense against untrusted foundries, capable of thwarting the insertion of even the stealthiest hardware Trojans. DR is directed and routing-centric: it prevents foundry-side attackers from connecting rogue wires to security-critical wires by shielding them with guard wires. Unlike shield wires commonly deployed for cross-talk reduction, DR guard wires present an additional technical challenge: they must be tamper-evident in both the digital and analog domains. To address this challenge, we present two different categories of guard wires: natural and synthetic. Natural guard wires are comprised of pre-existing wires that we route adjacent to security-critical wires, while synthetic guard wires are added to the design specifically to protect security-critical wires. Natural guard wires require no additional hardware and are digitally tamper-evident. Synthetic guard wires require additional hardware, but are tamper-evident in both the digital and analog domains.

ICAS: an Extensible Framework for Quantifying the Coverage of Defenses Against Untrusted Foundries

The transistors used to construct Integrated Circuits (ICs) continue to shrink. While this shrinkage improves performance and density, it also reduces trust: the price to build leading-edge fabrication facilities has skyrocketed, forcing even nation states to outsource the fabrication of high-performance ICs. Outsourcing fabrication presents a security threat because the black-box nature of a fabricated IC makes comprehensive inspection infeasible. Since prior work shows the feasibility of fabrication-time attackers' evasion of existing post-fabrication defenses, IC designers must be able to protect their physical designs before handing them off to an untrusted foundry. To this end, recent work suggests methods to harden IC layouts against attack. Unfortunately, no tool exists to assess the effectiveness of the proposed defenses---meaning gaps may exist.
This paper presents an extensible IC layout security analysis tool called IC Attack Surface (ICAS) that quantifies defensive coverage. For researchers, ICAS identifies gaps for future defenses to target, and enables the quantitative comparison of existing and future defenses. For practitioners,

ICAS enables the exploration of the impact of design decisions on an IC's resilience to fabrication-time attack. ICAS takes a set of metrics that encode the challenge of inserting a hardware Trojan into an IC layout, a set of attacks that the defender cares about, and a completed IC layout and reports the number of ways an attacker can add each attack to the design. While the ideal score is zero, practically, our experience is that lower scores correlate with increased attacker effort.