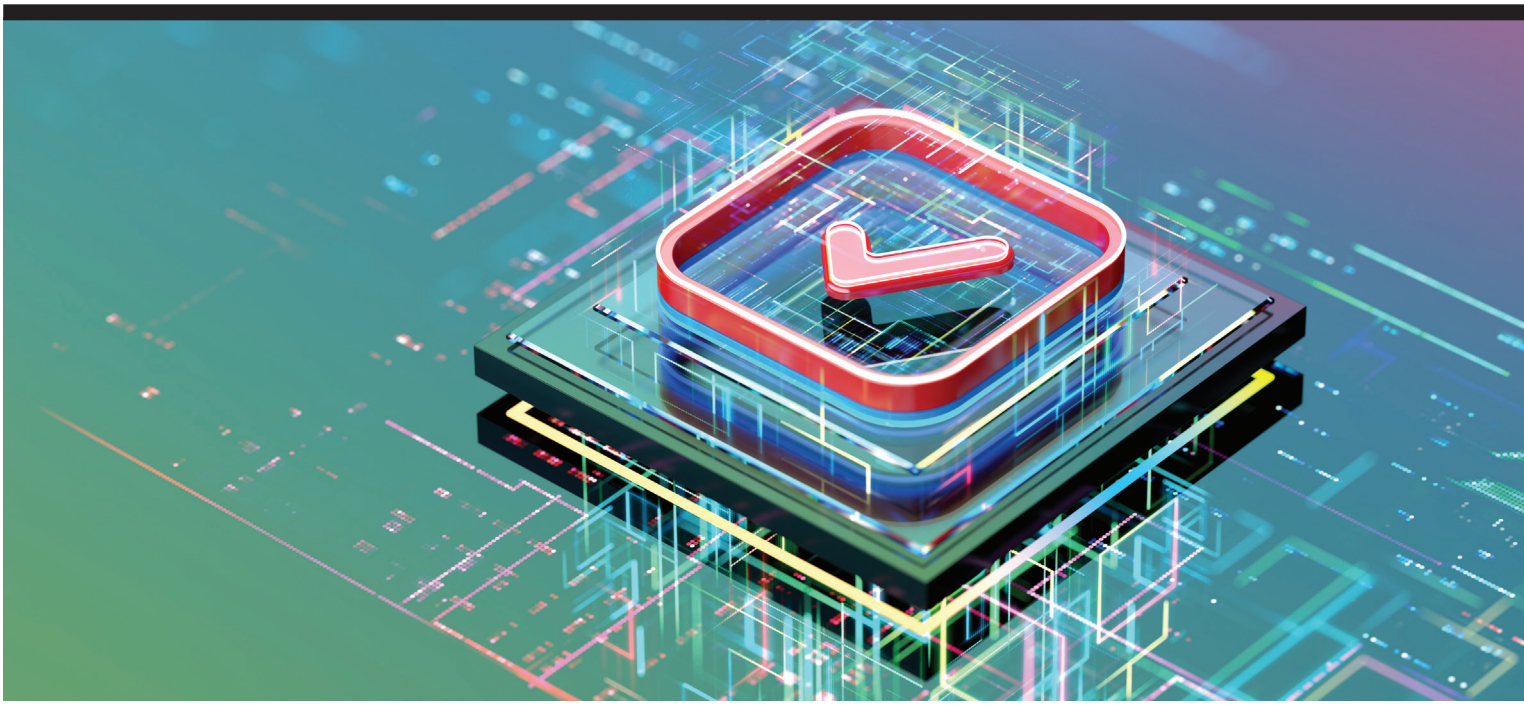
Pavel Frolikov, Youngil Kim, Renascence Tarafder Prapty, Gene Tsudik
University of California, Irvine, CA, USA



Resilient Software Integrity for IoT Networks

Excerpted from "TOCTOU Resilient Attestation for IoT Networks" from *SenSys '25: Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems* with permission. https://dl.acm.org/doi/10.1145/3715014.3722057 ©ACM 2025

Internet-of-Things (IoT) devices have become increasingly pervasive across modern society, often performing safety-critical functions. Network attestation – verifying the software state of all devices in a network – is a promising mitigation approach. However, current network attestation schemes suffer from lengthy Time-Of-Check-Time-Of-Use (TOCTOU) vulnerability windows, high latency, and interference from compromised devices. In this paper, we present *TRAIN*, a TOCTOU resilient attestation technique that synchronizes network-wide attestation, enables constant-time per-device reports, and remains resilient even when multiple devices are compromised.

IoT devices have become very popular in both consumer and industrial sectors. Some IoT deployment settings involve large numbers of interconnected devices that collaboratively perform sensing and actuation tasks in diverse applications domains, including smart factories, warehouses, agriculture, and environmental monitoring. When the nature of the devices' mission is safety-critical and/or involves collecting sensitive data, it is essential to protect such IoT devices against remote compromise.

A compromised sensor or controller can corrupt measurements, trigger incorrect actions, or disrupt physical processes. Operators, therefore, need a means to check whether the IoT devices upon which they rely are in a known benign state.

Remote Attestation (RA) provides one such check. A typical RA protocol has 4 steps: (1) trusted verifier challenges a remote untrusted device (prover); (2) prover generates an unforgeable attestation report based on cryptographic evidence of its current software state; (3) prover replies with the attestation report; (4) and verifier checks the report and decides on prover's state. RA is effective when the verifier checks one device/prover. However, naively applying the same idea to a potentially large network of IoT devices is inefficient and can produce misleading results.

To this point, Network Attestation (NA) aims to verify many devices (ideally) at once. Its goal is to collect individual device reports and allow the verifier to determine whether

Illustration, istockphoto.com

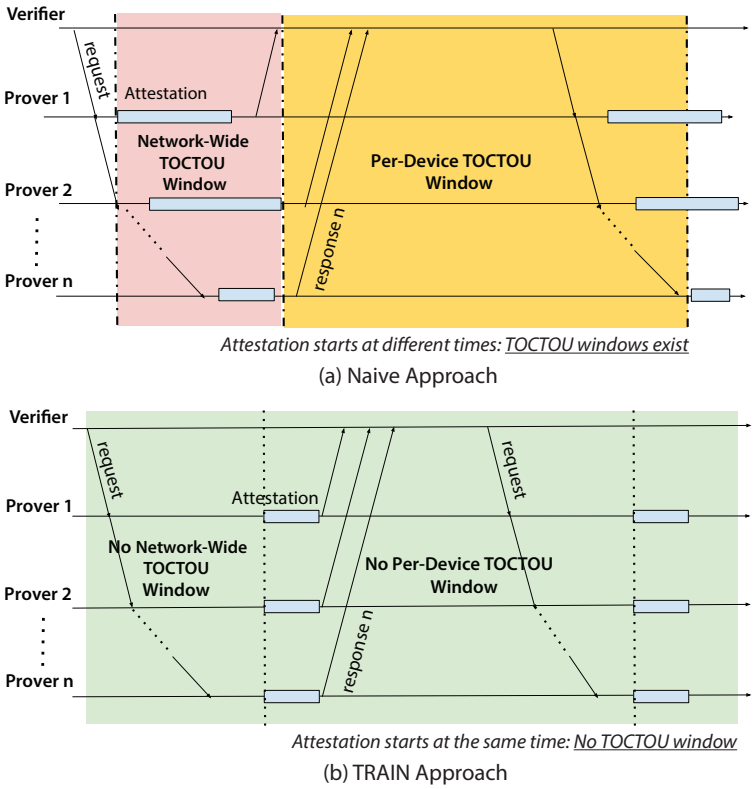


FIGURE 1. TOCTOU Window Minimized by TRAIN.

the network, as a whole, is trustworthy. This distinction is important; a network-wide security check should describe the state of the network at a meaningful point in time. If devices are checked at different times, the verifier may receive reports that are individually accurate, yet collectively misleading. Consider the verifier checking two devices. Device A is checked first and reports that it is benign. Shortly afterward, Device A becomes compromised. Due to network delay, Device B is checked later and also reports that it is benign. When the verifier receives Device B's report, it will incorrectly conclude that both devices are currently benign. Although both attestation reports are valid, they were produced at different times. Together, they do not provide a coherent snapshot of the network state.

TOCTOU: This timing gap is a network-wide Time-Of-Check-Time-Of-Use (TOCTOU) window. Traditionally, in the conventional single-device setting, TOCTOU refers to the period between two successive attestation checks, during which the device's software state is unknown and potentially compromised and undetected. Transient malware can exploit this gap by infecting a device, executing briefly, and then erasing itself before the next round of attestation. Figure 1 illustrates both issues: (1) per-device TOCTOU window between attestation rounds, and (2) network-wide TOCTOU window caused by devices being attested at different times.

Performance Overhead: Making NA practical for low-end IoT devices is challenging for several reasons. First, attestation itself is not free; computing a cryptographic report over device memory consumes time and energy, which matters for both battery-powered devices and safety-critical systems with tight timing constraints. Second, compromised devices may try to interfere with the attestation process by delaying, dropping, or modifying attestation messages. Thus, a practical NA scheme must verify software integrity, coordinate when devices attest, keep per-device overhead low, and remain resilient if some devices are *already* compromised.

To address these challenges, we construct TRAIN: TOCTOU-Resilient Attestation for IoT Networks. TRAIN coordinates NA so that devices perform their attestation at nearly the same time. To do this reliably even on compromised devices, TRAIN has a small Trusted Computing Base (TCB) on each device. This TCB ensures that the attestation requests are received, forwarded, and acted upon on time, even if the rest of the device software is compromised. TRAIN's TCB also reduces the attestation cost by limiting each check to a small, trusted state, rather than computing a cryptographic measurement over its entire program memory.

BUILDING BLOCKS

We focus on resource-constrained devices built around low-end microcontrollers (MCUs). As shown in Figure 2, these platforms typically include a simple CPU, memory regions for code and data, and basic peripherals such as timers, sensors, and actuators. Software runs directly on "bare metal," with no memory management for virtual memory or isolation.

TRAIN can be built atop either of two complementary styles of low-end Roots-of-Trust (RoTs): RATA [2] or CASU [1]. Both address the per-device TOCTOU problem while keeping attestation inexpensive for small MCUs.

RATA is a passive RoT architecture that mitigates per-device TOCTOU with minimal additional hardware. It securely logs the last program memory modification time to a protected memory region called the Latest Modification Time (LMT), which

cannot be modified by software. A device's attestation report then reflects the integrity of its software state indirectly through the LMT. The key idea is that any modification to the software in PMEM would necessitate an update to the LMT. Thus, by attesting the LMT, RATA effectively attests the software state without reading the entire program memory. This minimizes computational overhead by attesting only a fixed-size LMT, plus the verifier's challenge, instead of attesting the entire software image.

CASU is an active RoT architecture that provides run-time software immutability and authenticated software updates. It prevents unauthorized modification of program memory and blocks code execution from data memory. CASU monitors several CPU hardware signals such as the program counter, write-enable bit, and destination memory address, then resets the device if any violation is detected. CASU inherently prevents per-device TOCTOU since the software image cannot be overwritten by malware.

TRAIN also relies on GAROTA [3], another active RoT architecture that guarantees the execution of small trusted tasks. GAROTA provides two hardware properties: "guaranteed triggering" and "re-triggering on failure". The former ensures that a particular event of interest always invokes the corresponding trusted task, while the latter ensures that, if trusted execution is interrupted for any reason, the device resets and the trusted task will be executed first at boot-time. GAROTA supports several variants depending on the triggering event. TRAIN uses two of them: NetTCB, which guarantees that commands received over the network are processed by trusted code, and TimerTCB, which guarantees that a predefined trusted task executes at the scheduled time.

LIGHTWEIGHT AUTHENTICATION OF REQUESTS

In NA, a single bogus request can be costly. If attestation requests are not authenticated on the device, an adversary can readily mount a denial of service attack whereby an adversary floods all devices with bogus requests, each of which causes all devices in the network to perform attestation and generate numerous useless replies. This issue is deceptively simple. The naïve approach to address the problem is for the verifier to send an individual attestation request to every device in the network, authenticated with each shared key. This is unscalable for obvious reasons. Yet another trivial method is to assume a separate group key that is used exclusively for authenticating attestation requests issued by the verifier. This is quite efficient since a simple MAC would suffice. However, a key shared among a potentially large number of devices in the network raises the risk of its eventual compromise, which would have unpleasant consequences.

Hash Chains: TRAIN uses hash chains [4] to authenticate attestation requests. Hash chains provide a secure, scalable, and efficient means of authentication. They are used in settings where one party needs inexpensive authentication to a multitude of receivers. They provide forward security and efficient verification, while offering relatively simple key management. Although hash chains suffer from some fragility in terms of synchronization and timing requirements, these issues are more palatable than those that stem from managing large numbers of shared keys.

DESIGN OVERVIEW

Network: TRAIN assumes a single verifier and a connected network of low-end devices. TRAIN is network-agnostic and can be realized over any popular medium (e.g., WiFi, Bluetooth, Cellular, Zigbee, Matter).

Remote Attestation Architecture for Provers: Each device performs either RATA-style or CASU-style attestation. With RATA, the device's report authenticates the protected LMT. In contrast, CASU prevents any program memory modification (except via secure update), thus obviating the entire need for remote attestation. CASU supports a "secure heartbeat" feature, whereby the verifier issues a random challenge, and the device simply computes (and returns) a keyed hash over that challenge. This costs about the same as attestation token computation in RATA.

Guaranteed Network Functionality: TRAIN relies on NetTCB and TimerTCB to protect message handling and timing. NetTCB places the primary network interface within TRAIN's TCB, ensuring that TRAIN messages are handled before ordinary application traffic, even though the rest of the device software is compromised. TimerTCB ensures that attestation begins at the scheduled time, rather than being delayed by untrusted software.

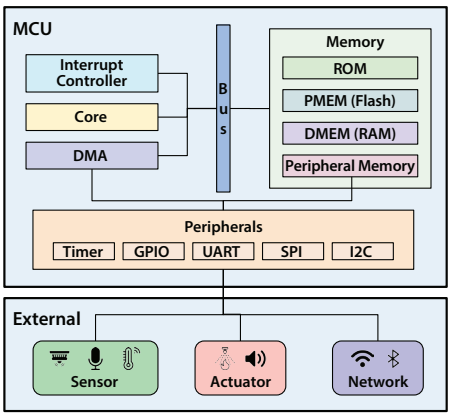


FIGURE 2. System Architecture of a Typical Low-end MCU.

PROTOCOL OVERVIEW

Commodity Real Time Clocks (RTCs) are now readily available for under $0.60 per unit. This affordability marks a significant shift from the past, when real-time security features were often too costly for IoT devices. TRAIN has two message types.

Message Types: TRAIN uses two attestation-specific message types: requests and reports. Incoming TRAIN requests or responses are processed by trusted code, and outgoing requests are generated by trusted code. TRAIN software prioritizes handling TRAIN protocol messages over other tasks. Non-TRAIN incoming messages are forwarded to the intended application outside the TCB, while ordinary outgoing messages are forwarded to their destination.

- **Attestation Request** tells devices that a network-wide attestation round has begun. It includes a packet type, the sender's identifier, a hash-chain value used to authenticate the request, and timing information that tells devices when to attest. The sender identifier also lets each device remember the neighbor from which it first received the request, thereby forming a spanning tree for forwarding reports back to the verifier.

TABLE 1. TRAIN Hardware Overhead

Architecture	Look-Up Tables	Registers
openMSP430	1854	692
CASU	1956	726
TRAINCASU	**1967 (+11)**	**740 (+14)**
RATA	1928	728
TRAINRATA	**1935 (+7)**	**737 (+9)**

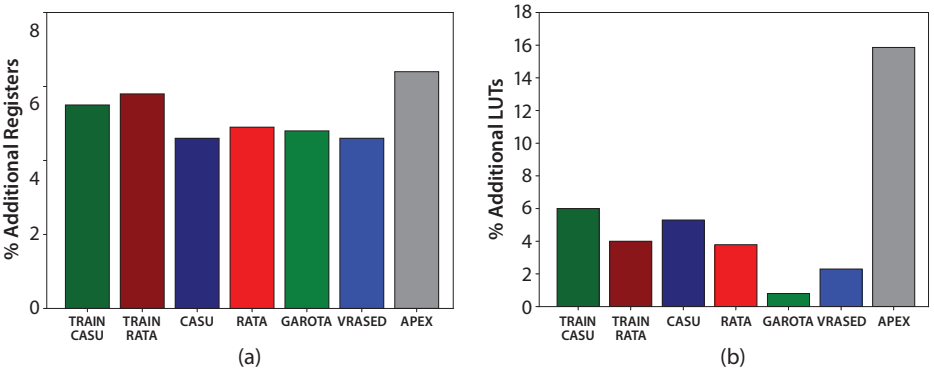


FIGURE 3. Hardware Overhead Comparison: (a) Additional Regs and (b) Additional LUTs.

- **Attestation Report** carries the result of the device's local check. For RATA-enabled devices, the report also includes the protected LMT value. It includes the device identifier, the hash-chain value from the corresponding request, the time at which attestation was performed, and an authentication token that protects the report's integrity. For CASU-enabled devices, the report instead acts as a secure heartbeat proving that the device is alive and responsive.

Network Attestation (NA): TRAIN performs network-wide attestation in three phases: Request Dissemination, Scheduled Attestation, and Report Collection.

Request Dissemination. The verifier begins an attestation round by broadcasting a single Attestation Request. Rather than sending a separate request to every device, TRAIN lets this request propagate hop-by-hop through the network of devices. When a prover device receives the request via NetTCB, trusted code verifies a hash chain to see if the request is authentic and fresh. If verification succeeds, the device records the neighbor from which it first received the request. The device records the neighbor from which it received the request as it is the return path to the verifier. Before broadcasting the request, the device replaces the sender identifier with its own identifier. This way, each downstream device learns where the request came from, and Attestation Reports can later flow back toward the verifier along the reverse path.

Scheduled Attestation. The request also carries the time at which attestation should occur. After forwarding the request, the prover arms a TimerTCB to trigger at that time. When TimerTCB triggers, every device performs attestation, regardless of when it originally received the request.

Report Collection. After attestation, each prover sends its own Attestation Report to the neighbor it recorded earlier. It also enters a short forwarding period, during which NetTCB accepts reports from downstream devices and forwards them along the same reverse path towards the verifier. Once this forwarding period ends, the device returns to its normal operation. The verifier collects incoming reports, checks if each one corresponds to the current attestation round, verifies its authentication token, and records which devices successfully responded. At the end of the round, the verifier outputs the set of devices that attested successfully, failed verification, or did not report.

FORMAL ASSURANCE

Computer-aided formal verification is the process of proving that a system satisfies a set of formally specified correctness properties. TRAIN's security depends on a small amount of trusted logic behaving correctly even when the rest of the device software is compromised. To increase confidence in this logic, we formally verify the design of TRAIN using linear temporal logic. The verified properties capture the expected behavior of the trusted hardware and interrupt logic: unauthorized modifications to protected program-memory regions trigger reset, execution is restricted to approved code regions, timer and network interrupt handlers execute atomically, and attempts to tamper with trusted timer or communication configuration are detected. Model checking is a formal verification technique that exhaustively explores all reachable states of a system model to determine whether specified properties hold in every possible execution. We translate the Register-Transfer Level (RTL) implementation to a Symbolic Model Verifier (SMV) [5] model and use the NuSMV model checker to validate these properties over all reachable states. This helps rule out subtle corner cases that are difficult to catch through testing alone.

IMPLEMENTATION

TRAIN is prototyped atop openMSP430 [6], an open-source implementation of TI MSP430 MCU, written in Verilog HDL. OpenMSP430 can execute software generated by any MSP430 toolchain with near-cycle accuracy. We extended both RATA and CASU architectures to support TRAIN. TRAIN software, responsible for processing TRAIN protocol messages and generating attestation responses, is housed in ROM. For cryptographic operations we use a formally verified cryptographic library, HACL* [7]. It provides high-assurance implementations of essential cryptographic primitives, such as hash functions and MAC-s. The synthesized design is then deployed on a Basys3 Artix-7 FPGA board for prototyping and evaluating hardware design.

EVALUATION HIGHLIGHTS

Hardware Cost: Recall that underlying hardware RoT for prover devices in TRAIN is either CASU or RATA with additional hardware support from GAROTA. Table 1 compares CASU and RATA with and without TRAIN support. TRAIN-enabled CASU implementation requires 0.46% more Look-Up Tables (LUTs) and 0.55% more registers over CASU. Similarly, TRAIN-enabled RATA implementation needs

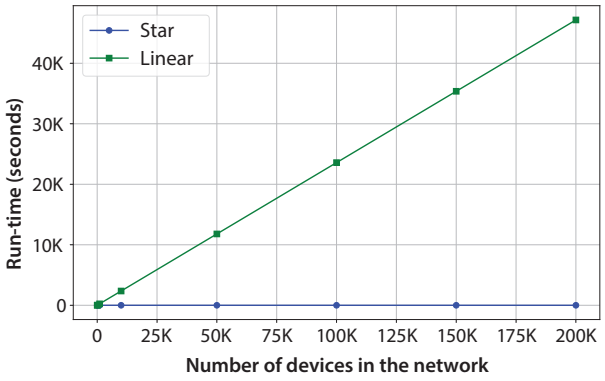


FIGURE 4. TRAIN Simulation for Line/Star Topologies.

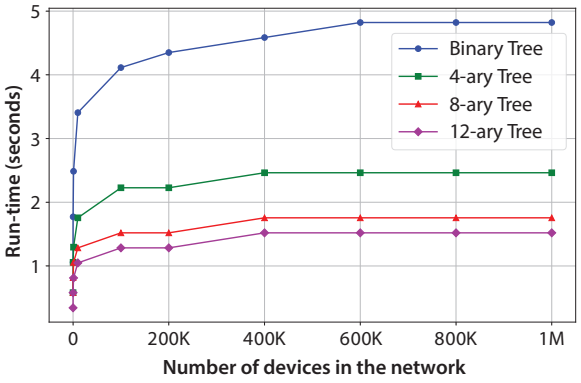


FIGURE 5. TRAIN Simulation for Various Tree Topologies.

0.05% LUTs and 0.69% registers over RATA. In both cases TRAIN adds fewer than 15 LUTs, implying minimal overheads incurred by NetTCB and TimerTCB.

Figure 3 compares TRAIN against other hybrid RoT architectures for low-end devices, including VRASED [8], RATA [2], CASU [1], GAROTA [3], and APEX [9]. Since these systems are all based on openMSP430, this gives a useful apples-to-apples comparison. TRAIN's overhead remains close to the underlying RATA and CASU baselines. In contrast, APEX incurs higher hardware overhead because it requires additional hardware support for generating proofs of execution.

Run-time Overhead: Since the verifier is not a resource-constrained device, we focus on the overhead incurred on the prover device. TRAIN's prover-side work consists of two main operations: verifying the Attestation Request and generating the Attestation Report. On an 8MHz MSP430-class device, TRAIN-enabled CASU takes 13.0ms to verify a request and 29.5ms to generate a report. TRAIN-enabled RATA is nearly identical, taking 12.9ms for request verification and 29.8ms for report generation. Each attestation instance takes only 42.5-42.7ms, while the total energy cost is also small; approximately 0.00221mWh per attestation instance.

Scalability: We conduct network simulations using the OMNeT++ [10]. Simulations are performed at the application layer. Cryptographic operations are simulated using delays that correspond to their actual execution times on TRAIN-enabled CASU and TRAIN-enabled RATA prover devices. Simulations are conducted with various spanning tree topologies: line, star, and several types of trees, with degrees ranging from 2 (binary) to 12. We also vary the number of devices from 10 to 1,000,000. Simulation results for TRAIN-enabled CASU and TRAIN-enabled RATA are almost identical, thus, only TRAIN-enabled CASU results are shown in Figures 4 and 5. The actual run-time for the star topology is 343ms. For a network with a tree topology, TRAIN run-time overhead is logarithmic in the number of prover devices since the tree height governs it. Simulation results show that TRAIN is efficient in both small and large networks with various topologies.

CONCLUSION

IoT deployments increasingly rely on networks of low-end devices to monitor and control physical environments. As these deployments grow in scale and importance, they also become attractive targets for compromise. TRAIN provides TOCTOU-resilient attestation for networks of low-end IoT devices by coordinating when devices attest. It facilitates simultaneous attestation across the network. TRAIN keeps runtime and energy overhead low by relying on a small trusted compute base and fixed-size attestation inputs. For additional details, refer to the full paper in SenSys 2025 [11]. ■

Pavel Frolikov is a PhD candidate at the University of California, Irvine. His research focuses on IoT security and privacy.

Youngil Kim received a PhD in Computer Science from UC Irvine in 2025. He is a Senior Firmware Security Engineer at Lucid Motors. He works on embedded system security and privacy, with an emphasis on secure architecture and hardware-assisted protections.

Renascence Tarafder Prapty earned a PhD in Computer Science from UC Irvine in 2025. Her research spans IoT security, network security and usable security. She is currently a Software Engineer at Cisco, working on data-plane and control-plane features for Open Source Network Operating System, SONiC.

Gene Tsudik is a Distinguished and ICS Alumni Professor of Computer Science at the University of California, Irvine. His research focuses on privacy, computer & network security, and applied cryptography.

REFERENCES

[1] I. De Oliveira Nunes, et al. 2022. CASU: Compromise Avoidance via Secure Update for Low-End Embedded Systems. *IEEE/ACM ICCAD '22*.

[2] I. De Oliveira Nunes, et al. 2021. On the TOCTOU Problem in Remote Attestation. *ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*, 2921–2936.

[3] E. Aliaj, et al. 2022. GAROTA: Generalized Active Root-Of-Trust Architecture (for Tiny Embedded Devices). *USENIX Security Symposium*, 2243-2260.

[4] L. Lamport. 1981. Password authentication with insecure communication. *Communications of the ACM* 24.11.

[5] K. McMillan. 2000. The SMV system. 10.1007/978-1-4615-3190-6_4.

[6] O. Girard. 2009. OpenMSP430: An open-source synthesizable MSP430-compatible micro-controller core. OpenCores. https://opencores.org/projects/openmsp430/

[7] J. Protzenko et al. 2017. "Hacl*: A verified modern cryptographic library, *ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*.

[8] I. De Oliveira Nunes, et al. 2019. VRASED: A verified {hardware/software} {co-design} for remote attestation. *USENIX Security Symposium*.

[9] I. De Oliveira Nunes, et al. 2020. APEX: A verified architecture for proofs of execution on remote devices under full software compromise. *USENIX Security Symposium*.

[10] OpenSim Ltd. [n.d.]. OMNeT++ Discrete Event Simulator. https://omnetpp.org/

[11] P. Frolikov, et al. 2025. TOCTOU-resilient attestation for IoT networks. *ACM SenSys '25*, 88-101.