Securing Two-Party Authentication of SCADA Commands in the Field using PUF-Encrypted Public-Key Signing

Kareem Ahmad*, Arman Allahverdi*, Vincent John Mooney III*†‡ and Santiago Grijalva*‡
*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA
†School of Computer Science, Georgia Institute of Technology, Atlanta, Georgia, USA
‡School of Cyber-Security and Privacy, Georgia Institute of Technology, Atlanta, Georgia, USA

***Abstract*—In this paper, we propose a hardware-assisted public-key signing architecture for securing Two-Party Authentication (TPA) of SCADA commands in the field. To secure private key storage on physically-accessible field devices, signing keys are encrypted using an AES-256 key reconstructed from an SRAM physical unclonable function (PUF) on an NXP LPCXpresso55S69 development board. We implement our proposed architecture using a prototype device and evaluate command round-trip times across wired and wireless networks in two variants of the TPA protocol—Edge Intercept Mode (EIM) and Control-Center Intercept Mode (CCIM). Our experiments compare TPA-only operation, plaintext public-key signing, and PUF-protected public-key signing using RSA and ECDSA keys. Our results show that control center intercept is approximately 55% faster than EIM, that ECDSA generally outperforms RSA with respect to latency in our prototype, and that PUF-protected signing introduces minimal additional latency. Relative to the unsigned TPA scenario, our hardware-assisted public-key signing increases latency by less than 8 ms on average for ECDSA signing and less than 25 ms on average when using RSA.**

***Index Terms*—Physical Unclonable Function, PUF, SCADA, Authentication, Two-Party Authentication, TPA**

I. Introduction

Power grid control systems increasingly rely on networked communication between operators, control centers, and remote field devices. This connectivity improves operational flexibility, but it also expands the attack surface for Supervisory Control And Data Acquisition (SCADA) commands. In particular, commands that affect field equipment should not rely solely on the authenticity of the issuing side, since both remote field access and lone-wolf insider threats can lead to unauthorized or unsafe actions [1]. For sensitive commands, Two-Party Authentication (TPA) provides an additional layer of protection by requiring approval from a trusted authorizer before the command is executed. However, the security of authenticated SCADA commands also depends on how cryptographic signing keys are stored and protected. Symmetric authentication mechanisms such as HMACs are efficient, but they introduce a practical deployment concern: any party capable of verifying a message must also possess the secret key. This expands the risk of key exposure and makes secure key storage especially important for devices deployed in remote or physically accessible environments.

In this work, we strengthen authenticated SCADA command execution by combining public-key signature schemes with hardware-based private key protection using physical unclonable functions (PUFs). In the literature, PUFs have previously been utilized to store symmetric keys to secure SCADA software updates [2] and to secure commands sent to distribution switches [1]. PUFs have also been used to secure firmware updates for IoT devices and embedded systems [3]. In our proposed design, each signing entity is assigned a public-private key pair, allowing signatures to be verified using public keys while keeping signing authority restricted to the private keys. On the field device, the private signing key is stored only in encrypted form. The symmetric key used to encrypt and decrypt the signing key is reconstructed from an SRAM-based PUF. Our design contributes to the field of PUF-based hardware security and secures the attack surface created by persistent signing key storage by introducing a hardware-based root of trust into the authentication architecture.

II. Background and Prior Work

A. Physical Unclonable Functions

A physical unclonable function (PUF) is a hardware security primitive that exploits the manufacturing variability of silicon devices to produce device-specific pseudorandom behavior, or equivalently, a hardware fingerprint [4]. The central concept of PUFs is that even when multiple devices are fabricated from the same architecture, uncontrollable microscopic process variations cause each individual device, or instance, to exhibit slightly different physical characteristics when operated. PUFs convert these small physical differences into measurable digital responses, thereby providing a hardware-rooted source of uniqueness that is not explicitly programmed into the device. From a security perspective, PUFs are appealing because they address a long-standing practical problem in cryptographic implementations: while cryptographic theory often assumes secure key generation and storage, these assumptions are difficult to realize physically. The use of a hardware-based root-

This work has been partially supported by the U.S. Department of Energy (DoE) Office of Cybersecurity, Energy Security, and Emergency Response (CESER) under Cybersecurity for Energy Delivery Systems (CEDS) Agreement Number #DE-CR0000055 to the Georgia Tech Research Corporation: GridLogic: Hardware/Software Codesign for Deep Grid Visibility and Security

of-trust for the generation and storage of cryptographic keys provides a solution to these assumptions without requiring the storage of plaintext cryptographic keys in non-volatile memory.

The PUF literature contains a broad range of constructions, but for integrated digital systems, the most relevant class is often that of intrinsic silicon PUFs, which derive their behavior from fabrication variations already present in transistors. Common examples include arbiter PUFs, ring oscillator PUFs, glitch PUFs, and SRAM PUFs [4]. These constructions differ in the physical quantity they measure, such as path delay mismatch, oscillation frequency mismatch, or power-up state, but all of these approaches aim to amplify microscopic random variations into reproducible device-specific hardware fingerprints. Current industry adoption of PUFs is quickly growing, particularly in the form of SRAM PUF IPs integrated into system-on-chip designs for secure key generation and device authentication without storing secrets in non-volatile memory, with many companies commercializing solutions for IoT, automotive, and secure embedded systems. In practical applications, PUFs undergo an initial provisioning phase called the *enrollment phase*, in which raw PUF responses are measured under controlled conditions and processed to generate stable helper data, enabling reliable reconstruction of the same device-specific secret during subsequent operation.

B. Two-Party Authentication of SCADA Commands

Prior work proposed the GridLogic TPA protocol [5] to enable Two-Party Authentication (TPA) of sensitive SCADA commands in the field. The work makes use of four primary components. **Issuers** create and send commands, **GridLogic Devices** receive and act upon commands, **Authorizers** approve or reject sensitive commands, and a **Validator** coordinates communication between components. **Issuers** and **Authorizers** are typically employees (people) while **GridLogic Devices** and the **Validator** are non-human, i.e., computer hardware and software.

In practice, the determination of what constitutes a sensitive command is a complex matter that would fall to the discretion of each utility. A simple classification could be based on whether a command operates or changes the state of a Field Device rather than simply reading sensor data. Regardless, determining command sensitivity and developing metrics to quantify it is beyond the scope of this work; the experiments in this work treat all commands as sensitive.

The prior work in [5] proposed two variants of the GridLogic TPA protocol distinguished by the location at which commands are intercepted and held for authorization. In Edge Intercept Mode (EIM), commands are intercepted at a GridLogic Device before execution, while in Control Center Intercept Mode (CCIM) commands are intercepted before leaving the Control Center. EIM theoretically requires fewer infrastructure changes compared to CCIM at the cost of higher latency. The high-level control flows of both TPA variants are shown in Figure 1. In EIM, the Issuer sends a command directly to the target GridLogic Device. The device then sends

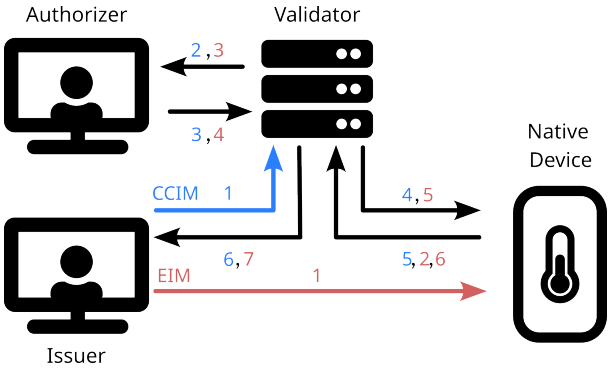

Fig. 1: GridLogic TPA Protocol. The EIM variant is shown in red and the CCIM variant in blue.

an authentication request to the Validator which checks if the command is sensitive. If the command is sensitive, a TPA request is sent to the Authorizer(s). When an Authorizer approves or rejects the command, the corresponding message is sent to the device which then executes or drops the command accordingly. Commands that are not sensitive are automatically approved by the Validator. In CCIM, the Issuer sends the command to the Validator first. If the command is sensitive, the Validator sends a TPA request as in EIM. If the command is rejected, the Validator simply drops the command. If the command is approved (or if it was not sensitive) the Validator sends the command to the target GridLogic Device with a signed approval. The device then executes the command.

C. System Architecture

We define our system from basic components. The definitions contained in this paper are similar to our prior GridLogic work [5] with minor changes that include the addition of an Administrator and a GridLogic Native Device. An example system is shown in Figure 2.

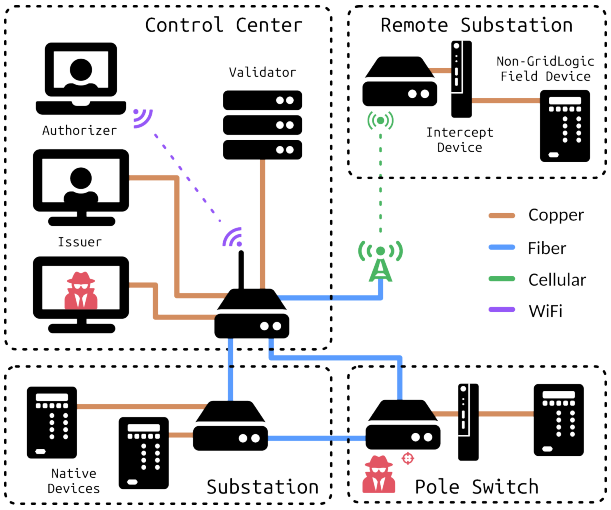

Fig. 2: Example System with Malicious Insider and Field-Station Attackers highlighted in red.

- **Issuer**: a potentially untrusted entity that initially creates and sends packets, typically an entry-level engineer or operator.
- **Field Device**: a power control device that receives and acts upon SCADA commands.
- **GridLogic Native Device**: a field device that receives SCADA commands using the GridLogic TPA protocol and performs actions based on the commands received.
- **GridLogic Interfacing Device, Adapter Function**: a device that enables non-GridLogic Field Devices to operate in a GridLogic-aware system. (Note that there exists a GridLogic Interfacing Device with In-Line functionality [6].)
- **Authorizer**: a trusted entity that reviews TPA requests and decides whether to approve or reject the corresponding command, typically a higher-level engineer or manager.
- **Administrator**: the highest level of trusted entity with the authority to register or unregister Authorizers, Issuers, or Devices with the Validator. The Administrator may also be an Authorizer, i.e., one person can serve in both roles.
- **Validator**: the central trusted entity that coordinates communication and TPA between Issuers, Devices, and Authorizers. The Validator may operate a Certificate Authority (CA) to centralize SSL certificate signing for components in the system [7].

D. Attack Surface

We focus our attack surface on two primary cases. First, we concern ourselves with a field-station attacker, who breaks into a remote pole-switch or field-station and directly plugs into the network. This attacker attempts to send disingenuous commands to other devices on the network. The second attacker is a low-level malicious insider: an engineer or operator with the capability to send commands from inside the control center and who possesses low-level signing keys. Please note that the malicious insider may also choose to attack in the field, and therefore we assume that a field-station attacker may also possess low-level signing keys. In both cases, we assume that the attacker does not have root access to the Validator or GridLogic Devices and acts without support from other individuals inside the organization. Finally, in both cases, the attacker may have hardware resources, software resources, or additional support from an external nation-state actor.

With respect to the PUF, we assume that enrollment is carried out by an Administrator and is therefore not subject to attack. Furthermore, collecting Challenge-Response Pairs in an attempt to characterize the PUF and taking advantage of error-correction PUF data stored in non-volatile memory requires root access or physical tampering with the device. Root access is already excluded from the attack surface. Physically tampering with the device requires more effort than reading non-volatile memory a few times, necessitating the removal of the device from the system for an extended period of time. We expect that a low-level engineer would not have the authorization to remove a power device from the premises and thus exclude this attack from the scope of this work.

III. Methodology

A. SRAM PUF Integration

In our experimental procedure, we utilize a PUF for the secure storage of cryptographic keys. More specifically, we use an NXP® LPCXpresso55S69 Development Board, which contains a dual-core Arm® Cortex-M33 microprocessor and an SRAM PUF. The specific SRAM PUF we use was designed by Intrinsic ID which is now part of Synopsys [8]. We configure the SRAM PUF to store a pre-determined 256-bit cryptographic key without requiring storage of the key in non-volatile memory during device operation. We implement our secure PUF-based key-storage mechanism by requiring the completion of an enrollment phase, during which the specific key value is defined and raw PUF responses are generated. Upon completion of the enrollment phase, the PUF-derived key is not explicitly stored in the non-volatile memory of the board; rather, the 256-bit key is encoded by the responses and helper data from the enrollment phase. Thus, during device operation, the key must be reconstructed from the device-specific enrollment phase outputs. Although the low-level algorithmic details of the reconstruction process are not specified by NXP, the PUF-based key reconstruction process on the LPCXpresso55S69 board is broadly based on the generation of a unique key code derived from SRAM startup data and device-specific SRAM variation characteristics [9].

B. Public Key Signing

Prior work uses SHA256-based HMACs for signing packets. However, HMACs rely on symmetric keys, allowing anyone capable of verifying a packet's identity to also forge a packet from that origin [5]. A better solution involves a situation where only the signer can generate a valid signature, and thus we transition to using public-key signing techniques. This is implemented as follows. Each entity capable of producing signed packets – Issuers, Devices, and the Validator – is assigned a public-private signing key pair. These keys may be either RSA or ECDSA keys [10]. When generating RSA keys, a 2048-bit key is chosen. When generating ECDSA keys, the NIST-standardized P-256 curve [11] is used. These keys are generated during enrollment and stored using the Privacy-Enhanced Mail (PEM) format. Public keys are securely registered in the Validator's database as described in Section III-C.

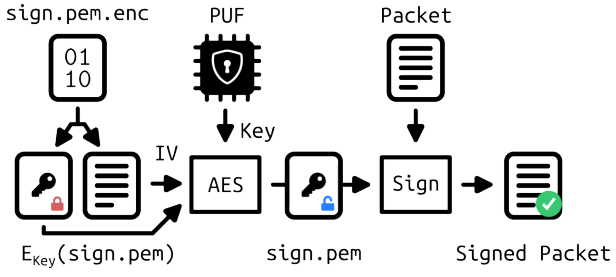

Fig. 3: Packet signing flow

The public key is stored in plaintext and shared across the network through the Validator. On Devices, the private keys are stored encrypted on disk using AES-256 in CBC mode with a PUF-derived key and random Initialization Vector (IV). To sign a packet, a Device reads the AES key from the SRAM PUF and decrypts the private signing key in memory. The packet is then signed using this private signing key. The private key is then erased from memory, and the signed packet is sent to its intended destination. The full signing flow is shown in Figure 3. Verifying a packet's signature only requires the public key associated with the packet's origin, which can be requested from the Validator.

C. Device Enrollment

Enrolling a new Device requires Administrator-level access. The process begins with assigning the device a hostname or IP address and installing the root certificate for the Validator's CA. All trusted SSL certificates in the network will be signed by this CA. The SRAM PUF on the LPCXpresso55S69 must undergo an enrollment phase before it can be used for secure key storage in our system architecture. During enrollment, firmware invokes the on-chip PUF controller, which derives a device-unique digital fingerprint from the SRAM PUF on the board and generates an *activation code* (AC). The AC contains error-correction data used to reconstruct the PUF digital fingerprint on subsequent device boots, but does not directly store the SRAM fingerprint. The AC is stored in nonvolatile memory in the system and is later supplied to the PUF controller to re-activate the PUF, enabling secure storage and reconstruction of keys on the LPCXpresso55S69 [9].

The next steps pertain to key generation. The Device generates signing RSA or ECDSA keys as described in Section III-B. The private key is immediately encrypted using the PUF key, and the plaintext copy is deleted. The public key is registered with the Validator using the Administrator's credentials. Then the Device generates a pair of SSL keys and a signing request. The request is sent to the Validator using the Administrator's credentials, and the Validator responds with a signed certificate. This certificate along with the private keys are used by NGINX to secure the HTTPS endpoint [12].

IV. Experiments

A. Network Architecture

To evaluate the performance overheads of public-key and encrypted-public-key signing, we set up the simplified model network shown in Figures 4 and 5. In the experimental network, the Validator runs on a Fedora workstation as a Flask [13] WSGI [14] application running on uWSGI [15] behind an NGINX [12] reverse proxy. We additionally construct a prototype temperature monitoring Native Device. This Native Device is constructed from the combination of a Raspberry Pi 5, Arduino UNO, NXP LPCXpresso55S69 PUF board, and TI LM95172 Temperature Sensor. The Raspberry Pi runs the uWSGI server and NGINX reverse proxy, and communicates with the Arduino and PUF boards over serial. The Issuer is a Python Command Line Interface (CLI) running on an Apple MacBook Air. The Authorizer and Administrator operate through the Validator's web interface secured by unique username and password credentials.

All network communication occurs over HTTPS using the Validator-signed SSL certificates. All devices are co-located within 20 feet of each other and connected over a low-usage network. The system is tested with connections over a 5 GHz WiFI access point and a 1 Gpbs Ethernet switch.

B. Command Timing

For a given command, we define the following timestamps:

- T_C : The time the packet is created
- T_A : The time the validator issues an authorization request
- T_E : The time the device starts executing the command
- T_D : The time the issuer receives response data for the command

From these, we define the roundtrip time T_R of a command as the total time between a command being issued and the device confirming the command's completion.

$$T_R = T_D - T_C$$

C. Experimental Scenarios

We measure roundtrip command time at four security levels: (i) Base, (ii) TPA, (iii) Sign and (iv) PUF. In the Base case, commands are sent directly from the Issuer to the Native Device without TPA or packet signatures. In the TPA-only case, TPA is enabled but signing is left disabled. In the Signing case, TPA and public-key signing are both enabled. Private signing keys are saved in plaintext on the device. In the PUF Signing case, TPA and public-key signing are both enabled. Private signing keys are saved in encrypted form and PUF keys are required for each signature. In all cases SSL is enabled with Validator-signed certificates.

We evaluate these four security levels in EIM and CCIM mode over both the wired and wireless networks. For each combined scenario, a temperature read command is sent 50 times in a loop with a two-second delay between commands.

To eliminate the variable delay of a human Authorizer deciding to approve or reject the TPA request, these performance experiments use an automated Authorizer script that approves every TPA request it receives.

V. Results and Discussion

The median roundtrip times are summarized in Table I. Box plots showing the distribution of roundtrip time across experiments is shown in Figure 6. From the experiments we observe three key findings. Firstly, we find that CCIM is an average of 55% faster than EIM. This supports the observation from our prior work that CCIM is more efficient, and is a result of CCIM having fewer communication steps than EIM [5]. This result may change in unbalanced networks if the GridLogic Device is significantly closer to the Issuer than to the Validator. Secondly, we find that ECDSA tends to outperform RSA in our tests. The difference is within the margin of error in the wired tests, but in the wireless tests,

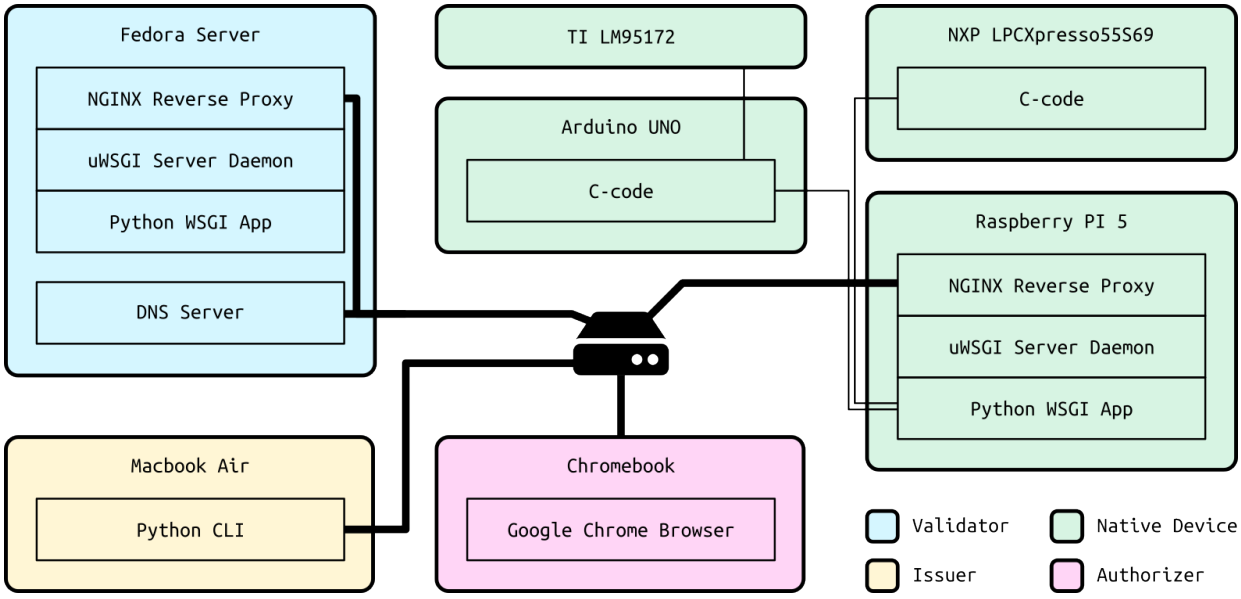

Fig. 4: Diagram of the Experimental Subsystem

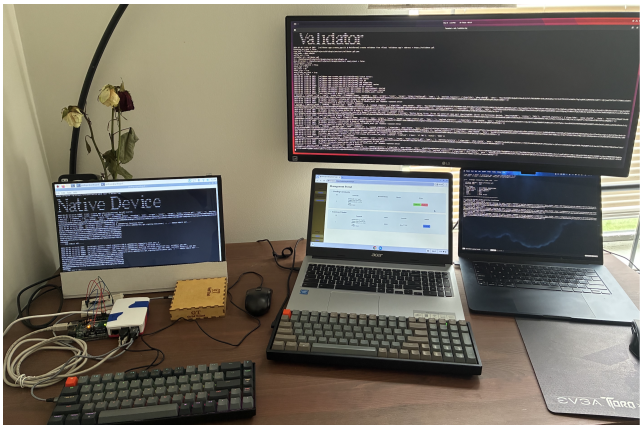

Fig. 5: Photo of Experimental Subsystem. From top to bottom, left to right: Validator, Native Device, Authorizer, Issuer

we find that using ECDSA can reduce roundtrip time by up to 50ms. Finally, we find that the overhead of encrypting the signing key using the PUF is small enough to be hidden by random jitter in our measurements of roundtrip time. In total, we find that encrypted public-key signing increases roundtrip latency by an average of < 8ms for ECDSA and < 25ms for RSA keys over the unsigned TPA case.

In terms of absolute changes to latency, the worst-case increase in roundtrip time was an average of 272ms over WiFi using EIM with RSA signing keys. Even in this case, the latency overhead incurred by the protocol itself is dwarfed by the latency that will be incurred by the human act of reviewing and approving or rejecting a command. Studies trying to measure the latency of human comprehension estimate the average time to press a button based on the semantic understanding of a word at 871ms [16]. Given this context, and that approving a TPA request will require reading more than a word, the protocol overhead of TPA and packet signing with encrypted keys is less than a third $271/871 \approx 31\%$ of the minimum human overhead. This is reduced in the equally secure case of using a wired network using CIM with ECDSA signing keys, where the protocol overhead is reduced to $129/871 \approx 15\%$ of the minimum human overhead.

TABLE I: Experimental Results

Network	Mode	Scenario	Algorithm	Roundtrip Time		
				ms	Δ	%
Wired	–	Base	–	13.0	–	–
Wired	CCIM	TPA	–	145.0	+132	1015%
Wired	CCIM	Sign	ECDSA	152.0	+139	1000%
Wired	CCIM	Sign	RSA	155.0	+142	1023%
Wired	CCIM	PUF	ECDSA	151.0	+138	992%
Wired	CCIM	PUF	RSA	153.0	+140	1008%
Wired	EIM	TPA	–	215.0	+202	1485%
Wired	EIM	Sign	ECDSA	234.0	+221	1631%
Wired	EIM	Sign	RSA	227.0	+214	1577%
Wired	EIM	PUF	ECDSA	234.0	+221	1631%
Wired	EIM	PUF	RSA	230.0	+217	1600%
WiFi	–	Base	–	60.0	–	–
WiFi	CCIM	TPA	–	179.0	+119	198%
WiFi	CCIM	Sign	ECDSA	186.0	+126	210%
WiFi	CCIM	Sign	RSA	193.0	+133	222%
WiFi	CCIM	PUF	ECDSA	182.0	+122	203%
WiFi	CCIM	PUF	RSA	203.0	+143	238%
WiFi	EIM	TPA	–	280.0	+220	367%
WiFi	EIM	Sign	ECDSA	275.0	+215	375%
WiFi	EIM	Sign	RSA	338.0	+278	463%
WiFi	EIM	PUF	ECDSA	282.0	+222	360%
WiFi	EIM	PUF	RSA	332.0	+272	453%

VI. Conclusion

In this paper, we introduced a PUF-assisted public-key signing architecture for securing Two-Party Authentication of SCADA commands in the field. Our design strengthens the prior GridLogic Two-Party Authentication protocol by replacing symmetric HMAC-based authentication with public-key signatures, allowing packets to be verified using public

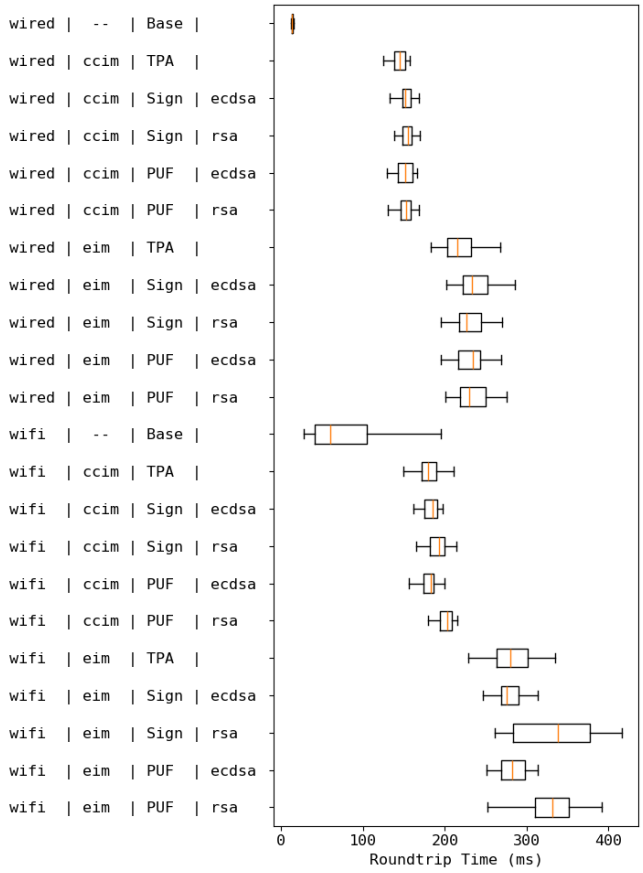

Fig. 6: Total Roundtrip Time

keys while restricting signing authority to protected private keys. To secure private key storage on physically accessible field devices, we encrypt each private signing key using a PUF-derived AES-256 key reconstructed from the SRAM PUF on an NXP LPCXpresso55S69 development board. We implemented our proposed architecture using a prototype GridLogic Native Device constructed from a Raspberry Pi 5, Arduino UNO, LPCXpresso55S69 PUF board, and temperature sensor integrated circuit. In our experiments, we evaluated TPA-only operation, plaintext public-key signing, and PUF-protected public-key signing over wired and wireless networks using both edge intercept mode and control center intercept mode. Our results show that authenticating at the control center is approximately 55% faster than at the edge, ECDSA signature latency generally outperforms RSA in our tested prototype, and the latency overhead of PUF-protected signing is small. Relative to the unsigned two-party authentication case, encrypted public-key signing increases latency by less than 8 ms on average for ECDSA and less than 25 ms on average for RSA.

Our results demonstrate that PUF-protected public-key signing can improve the security of SCADA command authentication without introducing significant latency into operator-driven two-party authentication workflows in our prototype system. Since the expected delay associated with human review and approval is substantially larger than the measured cryptographic and PUF-related overhead, our proposed architecture provides a practical path toward stronger authentication for field-deployed SCADA devices. We expect this to remain true in real-world SCADA systems with more complex networks, though absolute latencies will likely increase. Future work includes extending the prototype to additional SCADA command types, integrating the architecture with standardized field protocols such as DNP3, evaluating larger multi-device deployments, and analyzing the side-channel resistance of the PUF-assisted key reconstruction and signing process.

References

[1] J. Keller, K. Hutto, S. Grijalva, V. Mooney, T. Lewis, R. Barrett, B. Holland, and E. Patten, "Experimental system for supply chain cybersecurity of distribution switch controls," in *2024 IEEE Texas Power and Energy Conference (TPEC)*, 2024, pp. 1–6.

[2] S. Grijalva, V. Mooney, T. M. Lewis, A. King, A. Allahverdi, J. Keller, K. Hutto, R. Barrett, B. Holland, and E. Patten, "Experimental evaluation of a novel hardware-based authentication solution for securing electrical grids," *IEEE Transactions on Industry Applications*, vol. 62, no. 2, pp. 2250–2261, 2026.

[3] S. Falas, C. Konstantinou, and M. K. Michael, "A modular end-to-end framework for secure firmware updates on embedded systems," *J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 1, Sep. 2021. [Online]. Available: https://doi.org/10.1145/3460234

[4] R. Maes, *Physically Unclonable Functions: Constructions, Properties and Applications*. Berlin, Germany: Springer, 2013.

[5] K. Ahmad, A. Allahverdi, V. Mooney, and S. Grijalva, "GridLogic 2FA: Two-Factor Authentication for Critical Infrastructure Commands in the Field," in *2026 IEEE Texas Power and Energy Conference (TPEC)*, 2026.

[6] T. M. Lewis, G. Brown, A. J. Bush, J. Inniss, A. King, S. Grijalva, R. Barret, B. Holland, and E. Patten, "Deep Network Intervention: A Novel Semantics-Aware In-Line Device for Cyber-Threat Mitigation in Power-Control Networks," in *2026 IEEE Texas Power and Energy Conference (TPEC)*, 2026.

[7] R. Oppliger, *SSL and TLS: Theory and practice, third edition*, 3rd ed. Norwood, MA: Artech House, Jun. 2023.

[8] G.-J. Schrijen and V. van der Leest, "The competitive advantage of sram puf technology," https://www.synopsys.com/articles/competitive-advantage-sram-puf.html, 2025, accessed: 2026-04-20.

[9] NXP Semiconductors, "LPC55S6x: 32-bit Arm Cortex-M33 Microcontroller (Data Sheet)," NXP Semiconductors, Tech. Rep. Rev. 2.5, Aug. 2024. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/LPC55S6x.pdf

[10] N. I. of Standards and Technology, "Digital signature standard (dss)," U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 186-5, 2023.

[11] ——, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," U.S. Department of Commerce, Washington, D.C., Tech. Rep. NIST Special Publication (SP) 800-186, 2023.

[12] Igor Sysoev and F5 Inc, "nginx," https://nginx.org/, 2026.

[13] Pallets, "Flask," https://flask.palletsprojects.com/en/stable/, 2026.

[14] P. J. Eby, "Pep 3333 - python web server gateway interface v1.0.1," https://peps.python.org/pep-3333/, Python Foundation, Tech. Rep., 2010.

[15] uWSGI, "uWSGI," https://uwsgi-docs.readthedocs.io/en/latest/, 2026.

[16] O. Hauk, C. Coutout, A. Holden, and Y. Chen, "The time-course of single-word reading: Evidence from fast behavioral and brain responses," *NeuroImage*, vol. 60, no. 2, pp. 1462–1477, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S105381191200078X