GridLogic 2FA: Two-Factor Authentication for Critical Infrastructure Commands in the Field

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

***Abstract*—Due to its critical role in modern infrastructure, the power grid is becoming an increasingly popular target for cyberattacks. Despite this, many power utilities use legacy protocols with limited encryption support or leave encryption disabled, often out of concern for the latencies that would be incurred. Even in cases where utilities do encrypt traffic between the control center and remote field devices, utility infrastructure is vulnerable to insider threats. In this work we propose GridLogic 2FA, an http-based communication protocol that enables two-factor authentication of commands sent to field devices. We demonstrate that this protocol enables stopping malicious commands sent from a control center or remote attacker to an SEL 751 relay. Using this prototype, we find that the overheads range from 10% to 73% depending on the specific command and protocol variant. Additionally we empirically find that overheads incurred due to SSL range from 32% to 84%.**

***Index Terms*—SCADA, 2FA, two-factor authentication, SEL 751, GridLogic, Security, Power Grid, Relay**

I. Introduction

Power utilities are becoming increasingly common targets for cyberattacks due to their role as critical infrastructure. These cyberattacks often target the Supervisory Control and Data Acquisition (SCADA) systems used to control and operate power grid devices. Some of the most well-known attacks include the Russian cyberattacks on Ukraine's electric grid in 2015, 2016, and 2022 [1]. The first of these was able to successfully remotely operate breakers affecting nearly a quarter of a million customers for a few hours [1]. These attacks are only becoming more complex and prevalent with time. According to the latest 2025 report from Check Point Research, cyberattacks targeting utilities rose to an average of 1157 per week globally, up 42% from 2024 [2].

Despite the increasing prevalence of cyberattacks on utilities, many critical systems rely on legacy protocols that do not support encryption [3]. Many concerns exist with respect to encryption overhead and ensuring that safety critical operation is not impacted by improvements in cybersecurity.

This work provides three primary contributions. First, we propose the GridLogic* 2FA protocol enabling two-factor authentication of sensitive commands in the power grid. This protocol intercepts commands at devices in the field or in the control center and makes a 2FA request to a trusted individual if the command is determined to be sensitive. This can enable blocking the execution of malicious commands. Second, we demonstrate the blocking and security properties of this protocol using an SEL 751 relay. Finally, we empirically measure the overhead of enabling Secure Sockets Layer (SSL) encryption in a laboratory environment meant to mimic a utility context with and without GridLogic 2FA.

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

II. Background

A. SEL-751 Relay

The SEL-751 is a utility-grade Feeder Protection Relay from Schweitzer Engineering Laboratories that provides a handful of communication interfaces supporting DNP3, IEC 61850, and a custom ASCII protocol over serial [4]. This work makes use of the ASCII protocol for its relative simplicity for implementation and debugging. Connecting to the relay over serial provides a Command Line Interface (CLI) that allows testing commands and translates smoothly to sending commands from a python script. The CLI provides access to most of the Relay's functionality allowing users to read and change the relay's configuration and state. Executing a desired action ASCII protocol takes a few steps. First, one must elevate privileges using a set of ASCII commands and passwords to the level required to run the desired actions. Once the required access level has been obtained, one can send the ASCII commands required to complete the desired action. After the action is completed, one should downgrade the access level to the lowest level. We implement three high-level commands using this approach. The **open** command trips the relay causing the TRIP LED on the relay to turn red, producing an audible click as the contacts move, and changing the relay's status to TRIPPED. The **reset** command is the inverse of the open command, reverting the relay to the CLOSED state and clearing the TRIP LED. The **read** command reads the status of the relay. Executing each of these commands requires sending a fixed sequence of ASCII commands over the serial link. These commands each have different latencies and produce visible changes to the relay, making it easy to check whether or not a command was successfully executed.

B. Web-based Protocols

Since our GridLogic 2FA prototype uses Hypertext Transfer Protocol (HTTP)-based communication for demonstration, we include a discussion of the web-based protocols used. Integration with industrial SCADA protocols is discussed in Section VII. Representational State Transfer (REST) APIs are

built on top of HTTP and define a stateless interface to a web server [5]. A REST API request has three components: method, Uniform Resource Identifier (URI), and message body. The method is the HTTP method used for the request, most commonly GET (to request data) and POST (to send data). The URI is defined by the API and indicates the desired action. Finally the message body provides all data required for the web server to conduct the operation [5]. For example, **POST /command CommandBody** may be used to send a command to a device and **GET /command_sse** may be used to get information about running commands.

In cases where one needs a web server to push information to a device that is not a web server, a Server-Sent Event (SSE) stream can be used. SSEs are defined as part of HTML5 and readily supported by modern web browsers [6]. A connection to an SSE stream is initiated using a **GET** request to the stream's URI. The streaming connection between the client and server is kept alive until either side disconnects.

III. System Design

A. Definitions

The power grid system architectures explored in this paper include some or all of the following components:

- **Field Device:** A power control device in the field that receives SCADA commands and performs operations, e.g. a relay.
- **Interfacing Device:** An entity that supports the GridLogic 2FA protocol and connects over a known interface to a field device that does not natively support the GridLogic 2FA protocol. The interfacing device enables the non-GridLogic device to communicate with other GridLogic components and participate in 2FA.
- **GridLogic Device:** A Field Device that supports the GridLogic 2FA protocol or an Interfacing Device.
- **Issuer:** An entity that generates and sends commands. The issuer may be untrusted (e.g., a lone-wolf insider or a remote attacker).
- **Validator:** An entity that signs commands and coordinates 2FA between issuers, authorizers, and GridLogic devices.
- **Authorizer:** An entity that can approve or deny commands, acting as the second factor in 2FA command authentication. The authorizer must be trusted, typically a senior engineer or manager.

B. High-level Architecture

This work focuses on securing the communication between issuers and GridLogic devices as well as enabling two-factor authentication of sensitive commands. As such, without loss of generality, we limit our focus to subsystems consisting of a combination of GridLogic devices, issuers, authorizers, and a validator, henceforth referred to as communication subsystems. The entities in each communication subsystem must be connected over a network such that all GridLogic devices are reachable by all issuers and the validator is reachable by and can reach all other entities. Network topology and architecture have no restrictions outside of this requirement. For example, it is permissible for some GridLogic devices to be remotely connected to the network through a Virtual Private Network (VPN) tunnel over cellular telephony.

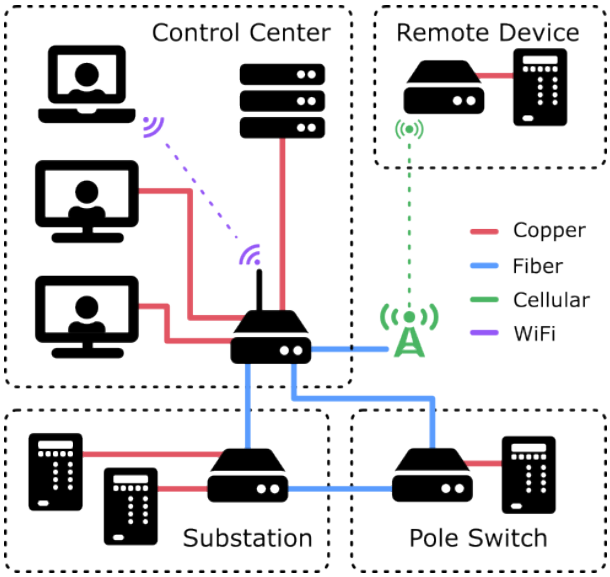

Fig. 1: Example network architecture

IV. GridLogic Protocol

The GridLogic 2FA protocol enables two-factor authentication of commands by intercepting commands at some point after being sent and before the commands are acted upon. At the interception point an authentication request is generated; the response determines whether to drop the command or allow it to proceed. We propose two modes of operation differentiated by interception point. Command interception occurs at the GridLogic device in Edge Intercept Mode (EIM), while in Control Center Intercept Mode (CCIM) interception occurs at the Validator before the command leaves the Control Center.

Communication in this protocol is handled by two REST APIs. A lightweight server on each GridLogic device provides a REST API for receiving commands and authorization information. A larger validator server provides a REST API to handle communications with issuers, authorizers, and GridLogic devices. The validator also provides an SSE stream for push notifications.

The rest of this section is organized as follows: Section IV-A describes the EIM variant of protocol, Section IV-B describes the CCIM variant of protocol, and Section IV-C delves into further detail about the security choices in the validator.

A. Edge Intercept Mode

In Edge Intercept Mode, command packets are sent directly from the issuer to the GridLogic device. The command packet contains information about the packet's origin, time of creation, intended destination, signature, and the command to be executed. The signature is a SHA256-based implementation of

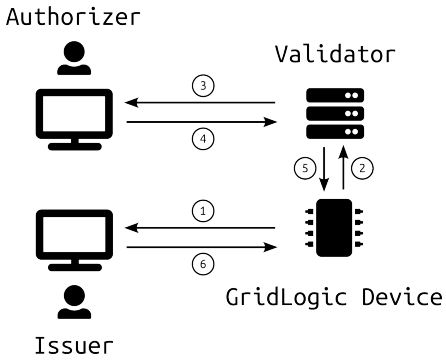

Fig. 2: Communication flow in EIM

the RFC 2104 HMAC construction [7]. Each issuer is assigned a unique signing key.

Upon receiving a command packet, the GridLogic device checks if the packet is well formed before conducting the following security checks. The device checks that the packet has not expired, was intended for this device, and that the packet's signature is valid and comes from the specified issuer. If all these checks pass, the device checks a set of local rules to determine if the packet needs a second factor of authentication. This is intended to reduce overheads on commands that do not require authentication, for example, certain sensor data reads. If the command requires authentication, the device forwards the packet unchanged to the validator.

Upon receiving a command packet from the GridLogic device, the validator checks if the packet is well-formed, has not expired, and contains a valid signature. If these checks pass, the validator then checks if the command requires authentication. If so, the validator responds to the device with a signed authentication pending message and sends a notification to all authorizers.

In our experiments, the primary interface between authorizers and the validator is through a web browser. Authorizers log into the validator's management interface using a username and password; the relevant security details are deferred to Section IV-C. Logging in presents the authorizer with a webpage containing a list of all commands requiring authorization and buttons to approve or deny each pending command. This webpage additionally connects to an SSE stream through which the validator pushes any new authorization requests or other information related to the execution of commands.

An approval or rejection from an authorizer triggers the following actions on the validator. First, the validator checks that the push request comes from an active authorizer session. If so, the validator checks if the command being approved or denied is still pending authorization. If so, a signed approval or rejection message is sent to the GridLogic device. The device repeats a set of security checks on the new packet, and if the checks pass and the command was approved, proceeds to execute the command. Upon completing the command, the device responds to the validator with a command completed message containing any relevant data if applicable (e.g., for a sensor read). The validator then forwards this response to all authorizers (and issuers) through the SSE stream.

B. Control Center Intercept Mode

CCIM tries to reduce overall latency by eliminating back and forth communication between the validator and the GridLogic device. In this mode, command packets—the same format as EIM—are sent by the issuer to the validator. The validator verifies the packet is well-formed, not expired, and has a valid signature. If the checks pass and the validator determines that the command requires 2FA, the validator sends a notification to all active authorizers through the SSE stream.

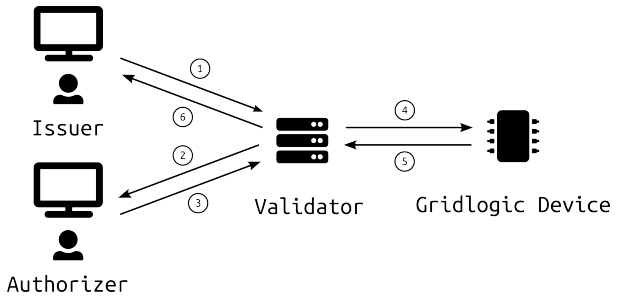

Fig. 3: Communication flow in CCIM

The web interface between authorizers and the validator is identical in CCIM to EIM. If a valid authorizer sends a rejection to the validator, the validator drops the rejected command packet. If, however, the validator receives an approval, it constructs and signs a validated command packet. This validated command packet contains all the information specified in the original command packet as well the authorizer decision. The validator then sends the validated command packet to the GridLogic device indicated in the original command packet.

Upon receiving a validated command packet, the GridLogic device checks that the packet is well-formed, not expired, intended for the device, was approved, and was signed by the validator. If all checks pass, the device executes the command and returns a corresponding response. Finally, the validator forwards the device response to all authorizers (and issuers) through the SSE stream.

C. Validator Security Considerations

The validator is the most complex component in the GridLogic 2FA system. It must support the management of multiple simultaneous connections, allowing multiple commands to pass through without blocking one another or terminating the SSE stream. This requires that the validator be capable of handling requests asynchronously while also maintaining data coherence.

The authorizer interface provided by the validator requires that access is limited to authorized authorizers. This is done using a username and password login system. Passwords are salted, hashed, and stored in a secure database [8]. When an authorizer tries to login, the submitted password is salted, hashed, and compared with the stored hash. If there is a match, a session is created and a signed encrypted cookie is returned along with the contents of the management webpage. Future communications from the authorizer to the validator must include this cookie to authenticate the authorizer's messages.

V. Experiments

A. Experimental Subsystem

We implemented a hardware prototype subsystem consisting of an interfacing device, a validator, an issuer, a web-based authorizer, and an automated authorizer. A photo of the prototype is shown in Figure 4.

Fig. 4: Prototype Subsystem. From left to right:
Issuer, Automated Authorizer, Validator, Interfacing Device, Relay

The interfacing device is implemented with a Dell Mini PC running Fedora Linux. The interfacing device is connected to an SEL 751 Relay over a serial-to-USB cable. The interfacing device runs a python Web Server Gateway Interface (WSGI) app implemented using Flask [9] and served by uWSGI [10] behind an Nginx [11] reverse proxy. The Nginx proxy provides both HTTP and HTTPS interfaces to enable measuring SSL overhead. The SSL connection uses 2048-bit RSA keys signed by a local Certificate Authority (CA). We implemented three commands on the interfacing device. The **open** command sends the ASCII commands required to trip the SEL-751 relay, the **reset** command resets the relay, and the **read** command reads and returns the current relay state as described in Section II-A.

The validator is implemented using a Macbook running a WSGI app served by uWSGI behind an Nginx reverse proxy. Similar to the device, the Nginx proxy provides HTTP and HTTPS interfaces. The SSL connection uses 2048-bit RSA keys signed by the same local CA. Password hashing uses the Scrypt key derivation function and a random salt [12], [13].

The issuer is a custom Python-based CLI running on a Dell Laptop running Fedora Linux. The issuer generates, signs, and sends commands in EIM or CCIM over either HTTP or HTTPS. The issuer can run interactively or execute a file.

The web-based authorizer is a Firefox web browser on a Dell Laptop running Windows 11. The browser is logged into the validator's manager portal. The web-based authorizer is used as the real-world authorizer. The automated authorizer is a python script that logs into the validator's manager portal and manages cookies to maintain the session. The automated authorizer waits for authentication requests and then automatically approves or rejects the request immediately. The automated authorizer is only used for measuring performance.

All the subsystem components are connected to the same network and configured with each other's IP addresses. The network is a wired network consisting of three Cisco IE3300 switches with all GridLogic components on the same subnet.

1) Base Case: The Base Case subsystem lacks two-factor authentication and all GridLogic security features. Thus it lacks both the validator and authorizer components. This is meant to provide a baseline comparable to real-world utility systems that do not include two-factor authentication. As shown in Figure 5, commands are sent by the issuer to the interfacing device, which, after checking that the packet is well-formed, sends the relevant serial commands to the SEL relay. Finally, the interfacing device reads the response data from the serial connection before responding to the issuer.

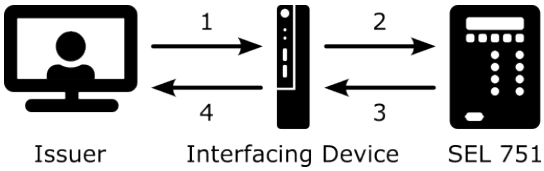

Fig. 5: Command flow in Base Case experiments

2) Edge Intercept: The edge intercept subsystem enables validation, authorization, and all security checks. The issuer is configured to send commands directly to the interfacing device as shown in Figure 6. Upon receiving a command, the interfacing device follows the EIM steps described earlier. If the device receives an approved response from the validator, it sends the ASCII commands required to execute the approved command to the SEL relay. It then reads the response from the serial bus and forwards it to the validator.

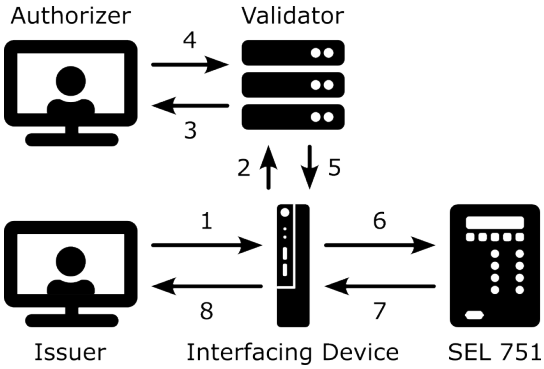

Fig. 6: Command flow in EIM experiments

3) Control Center Intercept: The control center intercept subsystem enables the validator, authorizer, and all security checks. The issuer is configured to send commands to the validator as shown in Figure 7. Upon receiving a command, the validator follows the CCIM steps described earlier. Similar to EIM, if the device receives an approved response from the validator, it sends the ASCII commands required to execute the approved command to the SEL relay and forwards the relay's response to the validator.

B. Security Experiments

To demonstrate the protections provided by the GridLogic 2FA protocol, we replace the issuer in each of the subsystem cases with a python script that manually creates and modifies command packets to see which modifications are detected.

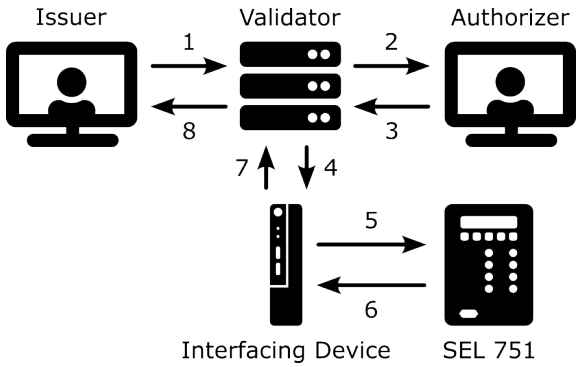

Fig. 7: Command flow in CCIM experiments

The modifications tested are shown in Table I. PASS indicates that the packet was automatically detected and dropped or rejected. FAIL indicates that the modification was not detected and the command was executed anyway.

TABLE I: Security Tests

Case	Base	EIM	CCIM	Detection Method
Malformed packet	PASS	PASS	PASS	Packet invalid
Origin changed	FAIL	PASS	PASS	Signature failure
Target changed	FAIL	PASS	PASS	Signature failure
Packet manipulated	FAIL	PASS	PASS	Signature failure
Forged command	FAIL	PASS	PASS	Missing signature
Forged approve	–	PASS	PASS	Signature failure
Malicious insider cmd	–	PASS	PASS	Authorizer reject

C. Performance Experiments

To measure performance overheads of the GridLogic 2FA protocol and SSL encryption, we set up the following experiment with an automated authorizer and an issuer script. We configure the automated authorizer to immediately approve all 2FA requests it receives. Additionally, we configure the issuer to execute the script in Figure 8 sending commands repeatedly in a loop while monitoring command roundtrip time. Command roundtrip time is defined as the time between the issuer starting to send a command packet and the issuer receiving the command completed response on the SSE stream. This experiment is repeated for each subsystem (Base, EIM, CCIM) with and without SSL enabled.

$$t_{roundtrip} = T_{complete} - T_{create}$$

```
repeat(40) {
    send relay open; sleep 0.5;
    send relay reset; sleep 0.5;
    send relay read; sleep 1.5;
}
```

Fig. 8: Issuer script

VI. Results and Discussion

A box and whisker [14] plot of the latencies of each of the three commands in the performance experiment is shown

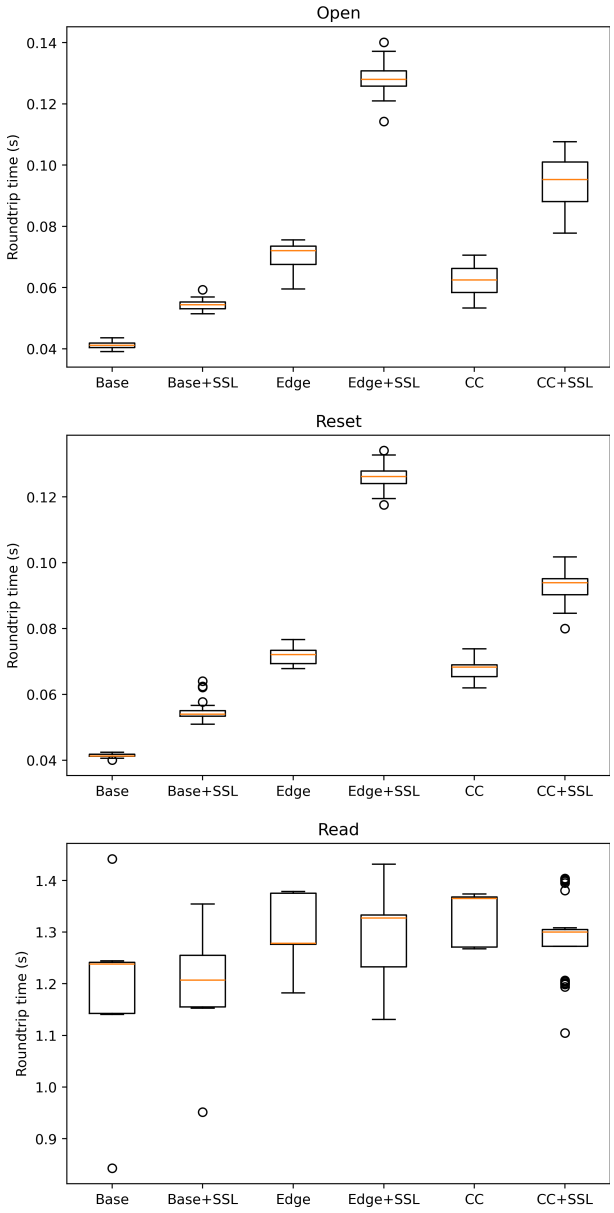

Fig. 9: Roundtrip time across modes and commands

in Figure 9. Note that the y-axis does not start at zero. Per-command performance overheads with and without SSL is shown relative to the Base Case in Tables II and III. For **open** and **reset** commands, we observe that CCIM has up to 40% lower protocol overhead (27% lower latency) than EIM. This is due to the reduced number of communication steps in CCIM. We expect this difference to increase in real-world systems where issuers are in relative proximity to the validator while devices can potentially be even miles away. We also find that while percentage overheads are much lower for the **reset** command, due to its higher base latency, the **reset** command also has larger variations in roundtrip time and higher absolute overheads due to larger data payloads.

Table IV shows the overhead of enabling SSL on each command. For the **open** and **reset** commands, SSL overhead ranges from 32% to 84% with the highest in EIM. Accurately

TABLE II: 2FA Protocol Overhead (SSL Disabled)

Command	Base	EIM			CCIM		
	ms	ms	Δ	%	ms	Δ	%
open	41	70	+29	70%	62	+21	52%
reset	41	72	+31	73%	68	+27	63%
read	1191	1307	+116	10%	1327	+136	11%

TABLE III: 2FA Protocol Overhead (SSL Enabled)

Command	Base	EIM			CCIM		
	ms	ms	Δ	%	ms	Δ	%
open	54	129	+75	139%	94	+40	74%
reset	54	126	+72	133%	93	+39	72%
read	1205	1285	+80	6.6%	1295	+90	7.5%

measuring the SSL overhead of the **read** command is difficult due to the highly variable roundtrip time visible in Figure 9.

To contextualize the overheads of the GridLogic Protocol, we compare the absolute protocol overheads against the overhead of human semantic understanding of written text. In their work on measuring lexical and semantic comprehension times, Hauk et.al. measure that mental processes of semantic understanding of a word begins within 200 ms of the appearance of a written word [15]. Additionally, they measure that reaction time to press a button based on the semantic understanding of said word ranges between 400 ms and 1500 ms with a mean of 871 ms [15]. The maximum overhead of the GridLogic protocol in our experiments was 136 ms for the **read** command with SSL disabled in CCIM. This maximum overhead is $136/871 = 15.6\%$ of the average human reaction time to semantic understanding of a word. This means that in a small-scale practical implementation of GridLogic, the human overhead of determining whether a given 2FA request should be approved or denied will dominate any protocol overheads.

VII. Future Work

For our future work we plan to explore alternative encryption techniques that can be added to or in place of SSL, and to conduct a more thorough security analysis of the GridLogic 2FA protocol. Furthermore, in settings with a high frequency of notifications, alert fatigue can become a security risk. This is especially recognized in medical settings like hospitals, where alert fatigue can result in ignored notifications and hasty judgments [16]. To alleviate this, future work will explore limiting authentication requests to only commands which are unexpected or sensitive in a given context. This will require developing metrics to quantify if a given command is potentially harmful or unexpected given the state of the power utility at the time and automatically approving commands that are below specified thresholds.

TABLE IV: SSL Overhead

Command	Base		EIM		CCIM	
	Δ	%	Δ	%	Δ	%
open	+13	32%	+59	84%	+32	52%
reset	+13	32%	+54	75%	+25	37%
read	+14	1%	-22	-2%	-32	-2%

Performance measurements in this work focused on sending a single command at a time. Future work will explore the effects of background traffic and the impacts of having tens to hundreds of devices communicating over the same network. Additionally, our methods for measuring command roundtrip time did not directly measure variations in command execution time. Measuring this in the future could help improve understanding of the observed overheads.

VIII. Conclusion

Our proposed GridLogic 2FA protocol improves security in the power utility context by enabling two-factor authentication of sensitive commands. We measure the overheads of this protocol to be significantly lower than the human reaction time to approve or reject the 2FA request. Additionally, we find that the overhead of enabling SSL ranges between 32% and 84% depending on the number of communications and size of payloads. We believe these results should be useful data points to utilities considering enabling SSL in their systems. Utilities will additionally need to consider their own specific timing requirements, network architecture, and operational constraints.

References

[1] B. E. Humphreys, "Attacks on Ukraine's Electric Grid: Insights for U.S. Infrastructure Security and Resilience," May 2024. [Online]. Available: https://www.congress.gov/crs-product/R48067

[2] Check Point Research, "The State of Cyber Security 2025: Top threats, emerging trends, and CISO recommendations," September 2025. [Online]. Available: https://www.checkpoint.com/security-report/

[3] D. o. E. Office of Electricity, "Secure Communications Interoperability in the Power Grid," September 2023. [Online]. Available: https://www.energy.gov/oe/grid-communications-and-security

[4] Schweitzer Engineering Laboratories Inc., "SEL-751 Feeder Protection Relay," https://selinc.com/products/751/, 2025.

[5] M. Masse, *REST API Design Rulebook*. O'Reilly, 2011.

[6] WHATWG, "HTML Living Standard," Web Hypertext Application Technology Working Group, Tech. Rep., 2025. [Online]. Available: https://html.spec.whatwg.org/multipage/

[7] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Interet Requests for Comments, RFC Editor, RFC, February 1997. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2104

[8] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *HANDBOOK of APPLIED CRYPTOGRAPHY*. CRC Press, 1996, p. 390.

[9] Pallets, "Flask," https://flask.palletsprojects.com/en/stable/, 2025.

[10] uWSGI, "uWSGI," https://uwsgi-docs.readthedocs.io/en/latest/, 2025.

[11] Igor Sysoev and F5 Inc, "nginx," https://nginx.org/, 2025.

[12] C. PERCIVAL, "Stronger key derivation via sequential memory-hard functions," tarsnap, Tech. Rep., 2009. [Online]. Available: https://www.tarsnap.com/scrypt/scrypt.pdf

[13] C. Percival and S. Josefsson, "The scrypt password-based key derivation function," Interet Requests for Comments, RFC Editor, RFC, August 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7914

[14] R. McGill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978. [Online]. Available: http://www.jstor.org/stable/2683468

[15] 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

[16] P. K. Wan, A. Satybaldy, L. Huang, H. Holtskog, and M. Nowostawski, "Reducing alert fatigue by sharing low-level alerts with patients and enhancing collaborative decision making using blockchain technology: Scoping review and proposed framework (medalert)," *J Med Internet Res*, vol. 22, no. 10, p. e22013, Oct 2020. [Online]. Available: http://www.jmir.org/2020/10/e22013/