

Program on Technology Innovation: HTTPN

Securing Nuclear Network Communications in the Age of the Internet of Things

2016 TECHNICAL REPORT

Program on Technology Innovation: HTTPN

Securing Nuclear Network Communications in the Age of the Internet of Things

All or a portion of the requirements of the EPRI Nuclear
Quality Assurance Program apply to this product.

YES



EPRI Project Managers
R. King
M. O'Connor



3420 Hillview Avenue
Palo Alto, CA 94304-1338
USA

PO Box 10412
Palo Alto, CA 94303-0813
USA

800.313.3774
650.855.2121

askepri@epri.com

www.epri.com

3002008039

Final Report, December 2016

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THIS DOCUMENT WAS PREPARED BY THE ORGANIZATION(S) NAMED BELOW AS AN ACCOUNT OF WORK SPONSORED OR COSPONSORED BY THE ELECTRIC POWER RESEARCH INSTITUTE, INC. (EPRI). NEITHER EPRI, ANY MEMBER OF EPRI, ANY COSPONSOR, THE ORGANIZATION(S) BELOW, NOR ANY PERSON ACTING ON BEHALF OF ANY OF THEM:

(A) MAKES ANY WARRANTY OR REPRESENTATION WHATSOEVER, EXPRESS OR IMPLIED, (I) WITH RESPECT TO THE USE OF ANY INFORMATION, APPARATUS, METHOD, PROCESS, OR SIMILAR ITEM DISCLOSED IN THIS DOCUMENT, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR (II) THAT SUCH USE DOES NOT INFRINGE ON OR INTERFERE WITH PRIVATELY OWNED RIGHTS, INCLUDING ANY PARTY'S INTELLECTUAL PROPERTY, OR (III) THAT THIS DOCUMENT IS SUITABLE TO ANY PARTICULAR USER'S CIRCUMSTANCE; OR

(B) ASSUMES RESPONSIBILITY FOR ANY DAMAGES OR OTHER LIABILITY WHATSOEVER (INCLUDING ANY CONSEQUENTIAL DAMAGES, EVEN IF EPRI OR ANY EPRI REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES) RESULTING FROM YOUR SELECTION OR USE OF THIS DOCUMENT OR ANY INFORMATION, APPARATUS, METHOD, PROCESS, OR SIMILAR ITEM DISCLOSED IN THIS DOCUMENT.

REFERENCE HEREIN TO ANY SPECIFIC COMMERCIAL PRODUCT, PROCESS, OR SERVICE BY ITS TRADE NAME, TRADEMARK, MANUFACTURER, OR OTHERWISE, DOES NOT NECESSARILY CONSTITUTE OR IMPLY ITS ENDORSEMENT, RECOMMENDATION, OR FAVORING BY EPRI.

THE FOLLOWING ORGANIZATION, UNDER CONTRACT TO EPRI, PREPARED THIS REPORT:

Sandia National Laboratories

THE TECHNICAL CONTENTS OF THIS PRODUCT WERE **NOT** PREPARED IN ACCORDANCE WITH THE EPRI QUALITY PROGRAM MANUAL THAT FULFILLS THE REQUIREMENTS OF 10 CFR 50, APPENDIX B. THIS PRODUCT IS **NOT** SUBJECT TO THE REQUIREMENTS OF 10 CFR PART 21.

NOTE

For further information about EPRI, call the EPRI Customer Assistance Center at 800.313.3774 or e-mail askepri@epri.com.

Electric Power Research Institute, EPRI, and TOGETHER...SHAPING THE FUTURE OF ELECTRICITY are registered service marks of the Electric Power Research Institute, Inc.

Copyright © 2016 Electric Power Research Institute, Inc. All rights reserved.



Acknowledgments

The following organization, under contract to the Electric Power Research Institute (EPRI), prepared this report:

Sandia National Laboratories
1515 Eubank
Albuquerque, NM 87123

Principal Investigator
J. Baker

This report describes research sponsored by EPRI.

This publication is a corporate document that should be cited in the literature in the following manner:

*Program on Technology Innovation:
HTTPN, Securing Nuclear Network
Communications in the Age of the
Internet of Things.*
EPRI, Palo Alto, CA: 2016.
3002008039.



Abstract

Industrial instrumentation and control (I&C) network protocol, as well as common networking protocols, are generally insecure by design as functionality was the dominating design requirement. These protocols were not intended for use in the current and evolving threat environment. This has resulted in serious concerns about the security of critical infrastructure, especially those facilities that depend on inherently insecure protocols for daily monitoring and control. This problem is not being addressed by equipment vendors, as there is a lack of government regulation and end user understanding of the vulnerabilities. These issues, and a general need to be backward compatible with existing equipment, also result in minimal end user demand for changes toward a more secure networking environment.

Over the years, there have been security enhancements to existing protocols, but these changes often allow operation in an insecure manner to support backward compatibility. This further compromises the I&C and networking environment, and leads to extensive analysis and testing of production network configurations to determine the level of security implemented, and also often results in unusual implementations and manual procedures to meet required security needs.

The objective of this project is to describe the framework and implementations steps for a secure version of the current HTTPS (hypertext transfer protocol (secure)) layer protocol (HTTP over TLS (transport layer security)), and to determine the feasibility of creating a new, singular specification (herein called HTTPN) that would not be subject to backward compatibility issues. The concept would also employ state of the art cryptologic techniques to ensure authentication, authorization, and auditing of network transactions.

Keywords

Cyber Security
Cybersecurity
Secure Communication
HTTPS
HTTPN

Deliverable Number: 3002008039

Product Type: Technical Report

Product Title: Program on Technology Innovation: HTTPN: Securing Nuclear Network Communications in the Age of the Internet of Things

PRIMARY AUDIENCE: Information security and network engineering personnel at new nuclear generating facilities

SECONDARY AUDIENCE: General information security personnel

KEY RESEARCH QUESTION

The focus of this research was to determine the feasibility of creating a new, singular communications specification based on the existing HTTPS (Hypertext Transfer Protocol (Secure)) protocol (herein called HTTPN), that would not necessarily be subject to backward compatibility, and that would employ state of the art cryptologic techniques to ensure authentication, authorization, and auditing of network transactions. The report explores the use of HTTPN for use in communications, data transfer, and the possibility for use in equipment controlling applications.

RESEARCH OVERVIEW

Required compatibilities for HTTPN were first characterized based on existing specifications. Existing technologies and published solutions were researched and summarized. Public and private key management and distribution methods are examined for applicability. A gap analysis between the current specification and a future HTTPN specification is assessed. Finally, future work required to push the effort to an Internet Engineering Task Force (IETF) (or other) Request for Comments (RFC) specification is outlined. The report also provides the results of using a simulation test bed to test the concepts.

KEY FINDINGS

- The report describes the background of HTTP and TLS (Transport Layer Security), how they were developed, and for what purposes. This background provides the framework for describing a proposed enhancement for a new protocol, theoretically called “HTTPN,” that has inherently more secure features in a closed network environment
- The concept of HTTPN would not be constrained by backward-compatibility requirements of current Internet Protocol environments. Support for backward-compatibility often generated unintended security holes, and the lack of required backward compatibility for HTTPN is supported by the idea that a new nuclear plant could be considered an isolated ‘green field’ installation, where backward-compatibility is not necessarily required, and the security requirements are high enough to support any extra overhead.
- While described in theory, developing a secure network environment for nuclear plant infrastructure has a feasible path, but with some important considerations, such as:
 - Developing a specification could require infrastructure devices to meet certain performance benchmarks or have specific system timing requirements.
 - Hardware and component networking interfaces need to be able to handle a broader array of protocol stack and traffic, while not affecting any of the protocol layers below it, ensuring adaptability.

- Susceptibility and threat considerations need to remain at the forefront of any new secure environment, as even changes to the TLS/SSL (Secure Sockets Layer) protocols envisioned for HTTPN could be obsolete quickly.
- A roadmap for future work and deployment identified several items that should be addressed, including:
 - Timing requirements – specific device timing requirements and use cases for controlling data input and output, as well as efficiency
 - Communication characterization – defining communication attributes
 - Investigating other protocols for secure control
 - Investigation into protocol security technologies

WHY THIS MATTERS

Industrial instrumentation and control (I&C) network protocols, as well as common networking protocols, are generally insecure by design as functionality was the dominating design requirement. These protocols were not intended for use in the current and evolving threat environment, which has resulted in serious concerns about the security of critical infrastructure, especially those facilities that depend on inherently insecure protocols for daily monitoring and control. This problem is not being addressed by equipment vendors as there is a lack of demand, through insufficient technical government regulation, or end user understanding of the vulnerabilities. These issues, and a general need to be backward compatible with existing equipment, also results in minimal demand for changes.

HOW TO APPLY RESULTS

The information in this report can be used as a guide and roadmap for future research and development to deploy a secure network protocol, both for new power plants and for other operating infrastructure. The theoretical protocol concept of HTTPN is based on existing network protocol and makes several assumptions; however, the feasibility of implementing such a protocol has a realistic path forward. The Gap Analysis and Roadmap for Future Work in the report provides the reader resources to use should such a concept move forward toward development and deployment.

LEARNING AND ENGAGEMENT OPPORTUNITIES

- The Electric Power Research Institute's (EPRI's) Power Delivery and Utilization (PDU) Sector, and the Nuclear Sector's Digital I&C Implementation Group and Advanced Nuclear Technology program are actively engaged in network cyber security research

EPRI CONTACTS: Ron King, Program Manager, rking@epri.com; Matt O'Connor, Sr. Project Manager, mcoconnor@epri.com

PROGRAM: Advanced Nuclear Technology (ANT) 41.08.01

IMPLEMENTATION CATEGORY: Reference – Early R&D

Together...Shaping the Future of Electricity®

Electric Power Research Institute

3420 Hillview Avenue, Palo Alto, California 94304-1338 • PO Box 10412, Palo Alto, California 94303-0813 USA

[800.313.3774](tel:800.313.3774) • [650.855.2121](tel:650.855.2121) • askepri@epri.com • www.epri.com

© 2016 Electric Power Research Institute (EPRI), Inc. All rights reserved. Electric Power Research Institute, EPRI, and TOGETHER...SHAPING THE FUTURE OF ELECTRICITY are registered service marks of the Electric Power Research Institute, Inc.

Commonly Used Acronyms

This section provides acronyms for key terms as they are used in this report.

Acronyms and Abbreviations

AI	Analog Input
AP	Access Point
CA	Certificate Authority
DI&C	Digital Instrumentation and Control
DNP3	Distributed Network Protocol
EMI	Electromagnetic Interference
FISMA	Federal Information Security Modernization Act
HMI	Human-Machine Interface
HTTP(S)	Hypertext Transfer Protocol (Secure)
I&C	Instrumentation and Control
IETF	Internet Engineering Task Force
ICS	Industrial Control System
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LTE	Long Term Evolution
MAC	Message Authentication Check
NIDS	Network Intrusion Detection System
OS	Operating System

OT	Operations Technology
PCI-DSS	Payment Card Industry – Data Security Standard
PKI	Public Key Infrastructure
QoS	Quality of Service
RA	Registration Authority
REST	Representational State Transfer
RFC	Request for Comments
RFI	Radio Frequency Inteference
SSL	Secure Sockets Layer
TLS	Transport Layer Security
VoIP	Voice over IP
WAF	Web Application Firewall
W3C	World Wide Web Consortium

Table of Contents

Abstract	V
Executive Summary	VII
Section 1: Introduction	1-1
Application	1-2
Cyber Threats	1-2
Cybersecurity Properties.....	1-3
Market Status.....	1-3
Section 2: HTTPS Overview	2-1
HTTP Features	2-1
Transport Layer Security	2-2
Section 3: HTTPS Enhancement (HTTPN).....	3-1
HTTPN Features	3-1
Usage	3-1
Persistent Communication	3-2
Methods	3-3
TLS in HTTPN	3-4
Client Authentication	3-4
HTTPN Handshake.....	3-4
Section 4: Key Management	4-1
Public Key Pinning.....	4-1
Private Certificate Authority	4-2
Manual Configuration	4-2
Automatic Authentication	4-2
Section 5: Network Considerations	5-1
Media 5-1	
Quality of Service / Fault Tolerance	5-1
IT Infrastructure	5-2
Section 6: Feasibility.....	6-1

Section 7: Gap Analysis.....	7-1
HTTP Protocol Examples	7-1
TLS Protocol Examples	7-1
HTTP/TLS Hardening & Best Practices.....	7-2
Security Headers	7-2
Server Configuration	7-3
Section 8: Roadmap and Future work.....	8-1
Section 9: References.....	9-1
Appendix A: HTTPN Experiments.....	A-1
Measuring TLS Overhead.....	A-1
Testbed Overview	A-4
Testbed Setup	A-4
Sample Experiment.....	A-5
Experiment Analysis	A-5
Network Diagrams	A-6

List of Figures

Figure 2-1 TCP Handshake and HTTP GET and OK	2-2
Figure 2-2 TLS Handshake	2-4
Figure 3-1 HTTPN TLS Handshake	3-5
Figure 8-1 Estimated Timeline of Future Work	8-3
Figure A-1 GET of Homepage without SSL/TLS	A-2
Figure A-2 GET of Homepage with SSL/TLS.....	A-3
Figure A-3 Block Diagram of HTTPN Testbed	A-6
Figure A-4 HTTPN Testbed Circuit Schematic	A-7



List of Tables

Table 2-1 HTTP Methods	2-2
Table 3-1 Proposed HTTPN Supported Methods.....	3-3
Table 3-2 Proposed HTTPN Crypto Suite	3-4



Section 1: Introduction

Industrial instrumentation and control, or I&C, network protocols, as well as common networking protocols, are generally insecure by design as functionality was the dominating design requirement. Some of the protocols used today were designed as many as 25-30 years ago, and were not intended to be used in the current and evolving digital threat environment. This has resulted in serious concerns about the security of critical infrastructure, especially those facilities that depend on inherently insecure protocols for daily monitoring and control. This problem is largely not being addressed by equipment vendors as there is a lack of pressure to do so, either by regulation or end user demand (often due to a lack of understanding of the potential vulnerabilities). These issues, and a general need to be backward compatible with existing equipment, also results in minimal end user demand for changes.

Over the years, there have been security enhancement to existing protocols, but these changes often allow operation in an insecure manner to support backward compatibility. This further compromises the I&C and networking environment through increased complexity. This leads to extensive analysis and testing of production network configurations to determine the level of security implemented, and also often results in unusual implementations and manual procedures to meet required security needs.

Due to the level of backwards compatibility needed and desired by almost all industries, there has not been a significant drive to create more secure protocols without the cruft and baggage of existing ones. However, with the deployment of new nuclear plants, combined with a clearer understating of the potential security threat to critical infrastructure, there is an opportunity to develop protocols that are secure by design. This may result in protocols that are not necessarily backwards compatible, but the isolation and effective 'green field' of a new nuclear island provides both an opportunity to implement less backwards compatible systems, and also a clear incentive for a more secure infrastructure.

A secure version of an accepted protocol that did not exhibit backward compatibility vulnerabilities would provide both an example and an imperative to improve I&C network security ahead of an evolving threat. New nuclear plant deployment provides a unique opportunity combined with a real security need to move forward with these new protocols. Once developed and put into the public as open and endorsed protocols, it would open the ability for existing nuclear, other energy facilities, and non-energy critical infrastructure, to begin requesting equipment that meets the more secure specifications.

Application

The use case for a new protocol specification can be applied across multiple sectors within an I&C network. The application can cross between traditional information technology, or IT, and operations technology, or OT, sectors of a network environment. This includes, but is not limited to:

- Computer to Computer communications, bidirectional (i.e. moving data over the IT system)
- Computer to Controller communications, bidirectional (i.e. changing values on computer screen, that change is reflected to controller, relaying instrument metrics back to computer/auditing system)
- Controller to Device communications, bidirectional (i.e. relaying commands from Computer, auditing, testing, reporting instrument metrics)
- Computer to Device communications, bidirectional (i.e. maintenance, auditing and testing purposes.) This suggests the possibility of a future architecture where the traditional hierarchy is challenged, and the dependency of controllers is lessened.

Cyber Threats

Industrial control systems face a unique set of threats and attack vectors predicated on the equipment installed at the site. As digital devices become more feature-rich, they also become susceptible to the same threats as traditional IT networks.

There are many threats an ICS can face; these include:

- sabotage
- destruction
- theft
- loss of public confidence
- even loss of life

These kinds of attacks could come in many different vectors also seen in traditional IT networks. These vectors include:

- denying availability (ex. shutting down a water pump, disallowing cooling of a system)
- sending valid communication inappropriately, causing an undesired affect (i.e. resending previously captured commands)
- eavesdropping on communications jeopardizing sensitive information
- masquerading/impersonation (i.e. Man-in-the-Middle attacks)
- integrity manipulation (i.e. changing error messages to normal)

It's these attack vectors that drive cybersecurity properties when designing secure systems.

Cybersecurity Properties

When discussing cybersecurity problems, it's important to understand what each solution offers in terms of cybersecurity properties. Traditionally, the main three categories are known as the CIA triad – representing Confidentiality, Integrity, and Availability. The Federal Information Security Modernization Act, or FISMA, outlines these properties as such:

- Confidentiality, which means “preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information”
- Integrity, which means “guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity”
- Availability, which means “ensuring timely and reliable access to and use of information”

It's with these security objectives that cybersecurity solutions should be engineered. Unfortunately, proper design in one objective often comes at the sacrifice of another. For example, using the best cryptographic methods can strengthen a design's confidentiality, however, it may require a processing delay, creating a lower availability.

System architects must weigh these trade-offs and evaluate which properties are most important to the system's security as well as efficient operation when making decisions.

Market Status

Currently, there are a few shifting trends that would help the advancement of what this report proposes. First, the Internet of Things, or IoT, is growing faster than consumers (individual, commercial and industrial) can keep up with. Vendors are building in this ability, whether the consumer will use it or not. Connected devices are becoming more and more abundant everyday. This is opportune for ICSs looking to make that transition from analog to digital or to an IP-based communication network.

Secondly, HTTP is ubiquitous within the Internet, as practically everything that runs on the internet can speak it. Using representational state transfer, or REST, web services make communication on both sides much easier. REST allows developers to quickly tap into the HTTP methods offered. This is much easier than developing an application that translates data to the user or device.

Furthermore, computation and encryption is becoming cheaper. With a full computer now fitting in the palm of your hand, high computation at a low cost is more readily available than before. Additionally, engineers are getting smarter on

how to handle encryption, by offloading it onto separate chips, doing it in hardware, etc. For these reasons, devices can be more secure than prior generations of technology.

Section 2: HTTPS Overview

Hypertext Transfer Protocol, or HTTP, is a protocol standard used by computers to communicate data. HTTP is a request-response protocol between a client, typically a web browser (ex. Mozilla, Chrome) and a server (ex. Apache, lighttpd). In the most common case, the client requests a webpage from the server, and the server responds with the page.

The Internet Engineering Task Force, or IETF, and the World Wide Web Consortium, or W3C, jointly created the standards for HTTP, which amounted to RFC 2068 [1] in 1997. HTTP has gone through a few revisions since its introduction. RFC 2616 [2] allowed for more persistence in which the same connection could allow more HTTP requests. This eliminated a handshake and teardown for each request, and cut latency.

HTTP/2.0 [3], created in May 2015, changed how data was sent, cutting down latency by doing small compression techniques, but leaving HTTP syntax and semantics the same.

HTTP was not originally created with security in mind and sends all data in plaintext, allowing eavesdroppers visibility into the conversation. Additionally, there's no authentication mechanism built into HTTP or the underlying TCP it's using. As such, a secure HTTP, or HTTPS [4], was created. This leverages the security libraries of Secure Socket Layers, or SSL. SSL has gone through revisions itself, and currently has a new name of Transport Layer Security, or TLS. Specific features and details of TLS will be discussed in another section.

HTTP Features

HTTP typically depends on the Transport Control Protocol, or TCP, for reliability in communication to ensure integrity for traffic. HTTP begins after a TCP connection has been established, and then uses HTTP actions to do communication.

HTTP offers several commands that nodes use to communicate data. The two most common, GET and POST, are seen in normal web browsing. A web browser will send an HTTP-GET message to the server, fetching the webpage it is hosting. Figure 2-1, for example, demonstrates this interaction.

In this packet capture, we see the initial TCP handshake, and then an HTTP-GET request from 192.168.38.1 for the homepage of 192.168.38.152. The

purple packets in between packets 4 and 10 is the TCP packets for the data being transferred. In packet 10, we see server respond with “200 OK” message, indicating the file has been transferred. This is the simplest of examples, however, the most fundamental when it comes to web browsing.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.38.1	192.168.38.152	TCP	78	61116->8000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=158505349 T...
2	0.000104	192.168.38.152	192.168.38.1	TCP	74	8000->61116 [SYN, ACK] Seq=0 Ack=1 Win=20960 Len=0 MSS=1460 SACK_PERM=1 ..
3	0.000228	192.168.38.1	192.168.38.152	TCP	66	61116->8000 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=158505349 TSecr=101...
4	0.000474	192.168.38.1	192.168.38.152	HTTP	369	GET / HTTP/1.1
5	0.000601	192.168.38.152	192.168.38.1	TCP	66	8000->61116 [ACK] Seq=1 Ack=304 Win=30000 Len=0 TSval=10159924 TSecr=150...
6	0.001190	192.168.38.152	192.168.38.1	TCP	83	[TCP segment of a reassembled PDU]
7	0.001225	192.168.38.1	192.168.38.152	TCP	66	61116->8000 [ACK] Seq=304 Ack=18 Win=131744 Len=0 TSval=158505350 TSecr=...
8	0.001313	192.168.38.152	192.168.38.1	TCP	204	[TCP segment of a reassembled PDU]
9	0.001381	192.168.38.1	192.168.38.152	TCP	66	61116->8000 [ACK] Seq=304 Ack=156 Win=131584 Len=0 TSval=158505350 TSecr=...
10	0.001395	192.168.38.152	192.168.38.1	HTTP	740	HTTP/1.0 200 OK (text/html)

Figure 2-1
TCP Handshake and HTTP GET and OK

HTTP/2 supports nine total methods (verbs) that represent the actions in the protocol. Table 2-1 below summarizes these methods and their high level purpose within the protocol.

Table 2-1
HTTP Methods

Method	Purpose
OPTIONS	Asks server which methods are supported
GET	Retrieves resource from server
HEAD	Same as GET, without message-body in response
POST	Request the server accepts data from client
PUT	Request that the data be stored under the supplied Request-URI
DELETE	Request that the server delete the resource identified in the Request-URI
TRACE	Used to invoke loopback of message. For debugging purposes.
CONNECT	Used to convert connection to secure tunnel
PATCH	Used to partially modify resources

Transport Layer Security

As discussed earlier, HTTPS was created in response to the obvious demand for security in web browsing. Initially, this addition was referred to as SSL, however, the fourth revision of that came with a new name change, TLS. At the time of this writing, TLS 1.2 [5] is the most current version, with TLS 1.3 in the process of development.

Use of HTTPS, when properly implemented, gives users a lot more security principles not offered in standard HTTP. Referring back to security properties discussed in Section 1, TLS provides both confidentiality and integrity, but at the expense of additional overhead of some availability (i.e. the processing adds some delay). Confidentiality is provided by using symmetric cryptography on the transmitted data. Integrity is done by using public-key cryptography, which is used to verify to one party the other party is indeed who they say they are. Additional integrity is provided by including a message authentication check, or MAC, that additionally helps verify the origin of the message and the untampered contents.

For better or worse, TLS was written to support many methods for implementing these fundamental aspects of security. At the beginning of a connection, the client and server must agree on which key exchange scheme, cipher suite and which MAC type to use. This negotiation process can be seen in the first two messages between the client and server in Figure 2-2 below, as we look at the TLS conversation.

This flexibility in implementation can actually become a security problem. As weaknesses and vulnerabilities in modern cryptography are discovered, some of these choices in the negotiation process become overall unsafe to use. Some servers are configured to accept older versions of TLS to ensure they can be available for their clients; sacrificing security for availability. Further, export controls restrict the use of some encryption techniques with countries the U.S. government has deemed sensitive. To preserve connectivity with foreign clients in these countries, domestically hosted servers must support legacy encryption schemes.

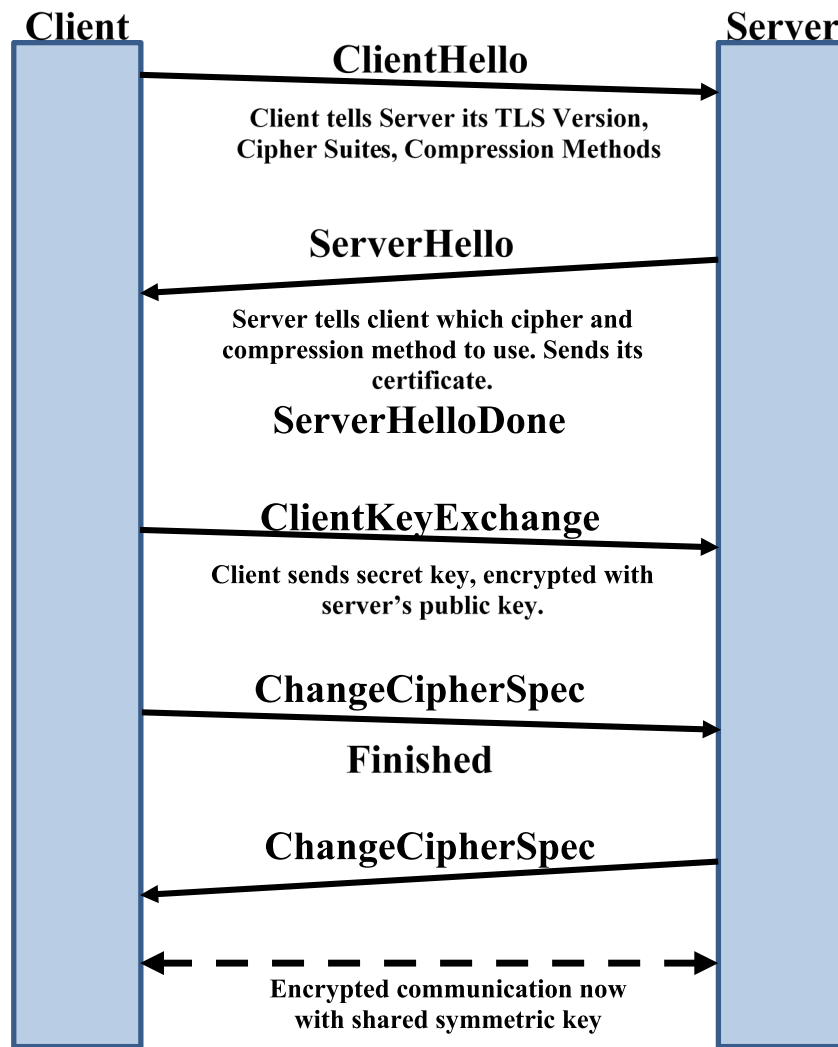


Figure 2-2
TLS Handshake



Section 3: HTTPS Enhancement (HTTPN)

Currently, it appears to be an opportunistic time where new nuclear power plants are being designed, as well as we are preparing for a future with “smart devices”. Because HTTP and other industrial I&C network protocols are not secure by design, there is a need for a protocol secure by design that could facilitate such an environment’s needs.

For an environment where security is crucial, sacrifices should not, in most cases, be made. Backwards compatibility and fallback to weaker ciphers are ideal for widely used situations, but when it comes to air-gapped, locally managed networks, having tighter requirements is much more reasonable of an expectation.

Conversely, the best security can come at the expense of time. In an environment where availability is imperative, the most secure cryptography methods may not be the best selection as they could impose an unacceptable delay.

It’s with these motivations that a singular, new specification, HTTPN, is being examined as a candidate to facilitate secure communications in industrial control systems, or ICS like a nuclear power plant (hence the “N” in the acronym). Suggestions for the most secure version of HTTPS are given, and then analyzed for feasibility.

HTTPN Features

Usage

The use case of HTTPN could be anywhere HTTPS is currently used. This would traditionally exist on the enterprise-like environment, consisting of mainly desktop computers. However, as we continue to see a growing industry of IoT devices, we see a potential for rapid deployment with little overhead assuming these devices can handle modern cryptographic computation. The goal is for any device in the ICS system, from the lower-level electronics to the operator workstations, to be able to utilize HTTPN.

In a dynamic, changing environment, HTTP methods can be utilized for more efficiency in transactions. However, in a more static environment, where a controller (server) is actively receiving data from end points (clients), the end points can POST data they want to send to the controller.

Persistent Communication

Inherently, however, there is a large problem in using HTTP as a control protocol. The trouble in this communication scheme is the intrinsic one-way communication in HTTP. Because HTTP is request-response, there's no easy way for the controller to give the end points commands within that same connection.

For services doing fixed, consistent communication, the keep-alive function within HTTP should be leveraged. First officially introduced in HTTP/1.1, this feature allows nodes to retain the connection between a client and server, without having the session reestablishment overhead of the TCP handshake and encryption for each unique data exchange. HTTP/1.1 considers all connections to be persistent, however, most Apache server versions have a default timeout. This parameter can be changed to meet a specific environment's need. Because of the nature of this, components in the network environment must be able to handle such connections. This means the controller (server) must be able to handle as many connections as end points (clients) its controlling.

Under the assumption that server initiated communication is needed, there are a few ways to address this issue.

1. Role reversal – each managed endpoint also acts as an HTTPN server, in addition to being a client. During the “push” of information to the controller, the endpoint is acting as a client. During the “poll” transaction, the controller is acting as the client, requesting services/status from the server (endpoint). This can stay within existing HTTP specification and permit bidirectional communication on-demand, without persistent keep-alive traffic.
2. Long Polling – the client sends an HTTP-GET request for a page the server is hosting. The server holds this request open until it has a command or message to deliver. The client retrieves this page and can execute this. The drawback of this scheme is unsynchronized requests and overall delayed time from request to execution. Simply put, this is a last resort for an environment that demands high availability like an ICS.
3. HTTP Streaming – At the beginning of connections, the client sends a request to the server. The server does not send a response until it has a command or new data. The request is never closed, and whenever there is new data, the server will respond. The drawback with this is always maintaining the connection in case the server has a message to send and the overall resource exhaustion that comes with this. This is a workaround of how HTTP normally operates, however, stays within HTTP specification.
4. Drastic changes in HTTP – Change the HTTP specification to allow at *least* half-duplex communication. Adding HTTP methods on the server side would allow for the server to be able to send commands as needed, without waiting or burning resources. A full-duplex solution is offered by a technology called WebSocket [6]. An upgrade to this mechanism is requested when communication begins, however, the server is not guaranteed to support such. Depending on the application, WebSocket may be enough

and could be mandatory in the HTTPN specification. Research is encouraged to ensure additional development into WebSocket is not needed.

WebSocket relies on the existing HTTPS setup and handshake with a slight addition. The client requests to the server that they upgrade the connection to WebSocket. If the server supports it, the server will respond back, accepting the upgrade. Bi-directional communication is now supported until one side closes the channel.

Notwithstanding, these options all have their own limitations as well as the shared restriction of relying on HTTP. Using a variant of HTTPS as a control protocol is definitely possible, however, it may not meet tight timing restrictions needed for crucial operation of an ICS, given the overhead associated with the workarounds available. With the advent of faster processors, though, overcoming the overhead may be possible. Ultimately, this is dependent on the system requirements and timing restrictions for the environment.

Issues with solutions 2 and 3 are addressed in RFC 6202 [7]. Maximal Latency was a large concern that is discussed for both of these. Other factors, such as overhead, resource allocation and incompatibility with networking intermediators are discussed.

RFC 6202 also gives an overview of existing technologies that allow asynchronous messaging from server to client. The Bayeux [8] protocol, for example, uses both long polling and HTTP streaming and also uses two independent HTTP connections. The most popular implementation of this is CometD, which has been successful in deployment for some specific applications.

Analysis into which of these technologies is best would require a more robust definition of the design requirements. Requirements for latency and availability, use case for connectivity and bandwidth needed should all be considered and evaluated when considering these options.

Methods

From the current understanding of how HTTPN would operate within an industrial I&C perspective, there are more features within HTTP that are simply unneeded that can be stripped away. Ideally, the only two HTTP methods needed for components to do what they need to do are HTTP-GET and HTTP-POST commands. Without necessity for other methods, these are the only methods that should be supported. An abridged table of the necessary methods for HTTPN is shown in Table 3-1.

*Table 3-1
Proposed HTTPN Supported Methods*

Method	Purpose
GET	Retrieves resource from server
POST	Request server accepts data from client

TLS in HTTPN

As discussed earlier, a large problem with HTTPS is the allowing of older versions of the protocol and weak cryptography. HTTPN would require a more hardened version of TLS, offering no TLS/SSL fallback and not allowing negotiation of cipher suites; the server would reject any crypto that is not in line with pre-approved settings.

TLS1.2 supports a wide array of key exchange methods, as well as ciphers and HMACs. As an example of a pre-selected crypto suite, the following recommendations are in Table 3-2.

Table 3-2
Proposed HTTPN Crypto Suite

Purpose	Algorithm
Key Exchange	DHE-256
Cipher	AES-256-GCM
Data Integrity	HMAC-SHA384

Other crypto methods should be considered if the proposed algorithms become obsolete or unsafe. HTTPN should be flexible in adapting new algorithms if that happens to be the case, but again should not allow negotiation of crypto parameters, but should be statically assigned at the time of provisioning.

Client Authentication

Referring back to Figure 2-2, the TLS handshake, we see that the server sends its certificate to the client to verify authenticity of the server. An optional step within TLS is for the client to offer its certificate to the server and the server to ensure its identity. This extra step will be enforced in HTTPN, requiring all parties to identify themselves. However, since the system is air gapped from public certificate authorities, a local certificate store must be utilized. During device enrollment in the system, their keys must be loaded into this identity management and key distribution mechanism. This is discussed largely in the following section.

HTTPN Handshake

Putting all of these suggestions together changes the TLS handshake, making it more detailed and specific. This is shown below in Figure 3-1.

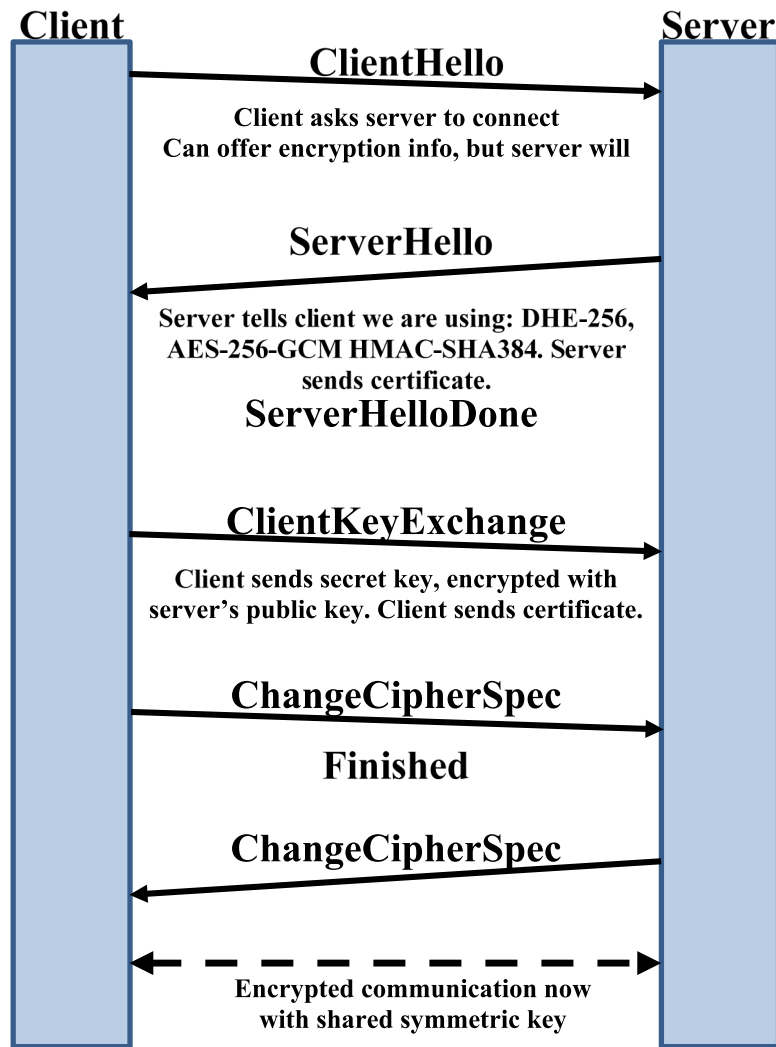


Figure 3-1
HTTPN TLS Handshake

Section 4: Key Management

Key management is the administration and organization of cryptographic keys for a system using cryptography. In a lot of scenarios, key management and distribution is the hardest problem for secure communications. Key distribution is part of the attack surface, so proper implementation is crucial. In order to continue discussing options for this, a quick overview of public key infrastructure, or PKI, must be discussed.

PKI is a construct used to organize public-key encryption. In this configuration, every entity has private and public keys. Only the entity knows its private key, and it uses that to sign messages it sends. Public keys are public, and are used to verify a message is signed from who it claims it is, providing integrity. Furthermore, a device can encrypt a message using the recipient's public key, and the recipient decrypts it using its private key, providing confidentiality. This environment is comprised of one or more of the following entities:

- Certificate Authority (CA) – the role of manager for certificate registering and creation
- Registration Authority (RA) – the role of validation of a specific certificate; typically, CAs act as RAs as well. For the purposes of this environment, we will use just CAs
- Subscriber/Relying Party – services and users of those services, respectively. In the context of this environment, all components will act as both in order to provide full authentication

In the public internet, there are hundreds of trusted CAs that act as validators of certificates, from both public and private entities. Lists of these CAs are typically preloaded into a web browser, and can be configured afterwards. This offloads the work of authenticity to other machines, however, it poses a problem as now this increases the attack surface as well as requires connectivity outside of your network – a problem for a potentially air-gapped network.

Public Key Pinning

The overall goal for designing this architecture can be summarized by the purposely paranoid axiom of “deny all, allow by exception”. This zero-trust philosophy can also help enforce communication policy that has been established for devices. This problem can be solved with a private centralized CA or be enforced at a lower level depending on the flexibility needed for the system. In either case, a technique called public key pinning can be used to harden these

communications. Pinning is the practice of associating a public key with a specific device. Using pinning can reduce the potential attack surface and reduce potential overhead by bypassing a CA. In a design in which a private CA is used, each device would need to pin the CA's public key.

Private Certificate Authority

Easier solutions can be achieved, however, when the entire environment is controlled, like an air-gapped network. One solution is to create a private, local CA that can operate as the authenticator for all devices. This allows for authentication without any traffic needing to go outside the environment. This is ideal for a large, dynamic private network, where different devices have various connectivity requirements.

Manual Configuration

A pedantic, yet effective method would be to load all necessary keys manually onto each device. As we saw in the previous method, having a CA increased administrative scalability in network changes. When working with a network that doesn't have many changes (i.e. new devices being added, communication policy changes, etc.) administrative scalability is less of a priority. In such situations, manually configuring certificates on the network may be preferred.

In this method, certificates are saved onto each device manually. There is a bit of overhead in initial configuration and setup. However, this allows for strict enforcement of network policies. By not adding certificates for machines that are not supposed to communicate with, disallowed communication cannot happen.

A large deterrent of configuration is the necessary labor in the case of the removal or addition of a device. Imagine a controller that controls two devices that also communicates with another controller, and a higher level controller. Replacing this controller would require manually configuring of all the devices it communicates with. This presents additional overhead for the system operators and potential downtime for the system. One-time enrollment/loading of key options may be available as third-party tools mature to better facilitate manual configuration.

Automatic Authentication

Future work could turn this manual process into an automated one. The idea of being able to insert a new device and it begin to work, *securely*, with no manual intervention is appealing, however, there are potential concerns.

The most difficult challenge would be developing vendor support. Devices could come preloaded with certificates signed by a vendor's private key. A CA-like terminal could then verify the device is legitimate, give the appropriate keys to it, assign it a new private key for the local network and the device could ideally work. This kind of authentication is done, but can be proprietary or specific to a vendor. Participation from all vendors (or design-exclusivity of a single vendor)

would be needed for this to work. Similar efforts are already done, however, the technology is not mature enough to guarantee security from supply-chain attacks. For example, this technique is used when enrolling VoIP desk phones in enterprise networks. Typically, a provisioning/roll out period is set and any future changes are processed manually.



Section 5: Network Considerations

While the discussion for a singular protocol continues, an additional conversation on the underlying network also seems appropriate. It's important to remember that shifting into an HTTPN-based world requires a network overhaul into an IP-based architecture. From transport medium to existing infrastructure appliances, much of the technologies and advances in the traditional IT realm are also available to an environment dedicated to HTTPN.

Media

Because HTTPS (and HTTPN) runs at the top of the networking stack, the specific media on the bottom of the stack is somewhat insignificant. With that said, however, discussion of the available mediums and their strengths/weaknesses should be had. Older twin copper cables carrying serial won't be sufficient, especially at the availability needed.

Modern 100BaseT Ethernet should be sufficient for proper function of an HTTPN based network, however, scalability and availability of a specific network may dictate 1000BaseT. Fiber cabling could prove to be a viable solution for some environments, and in fact, many new nuclear plants are installing fiber as part of construction.

A large share of the IoT devices coming into market are largely centered on their wireless capabilities. This is often desired for the ease of setup, reduced capital costs due to reduced wiring and cable trays, and lack of infrastructure changes when adding to a completed design. Additional network access points, or APs, would be necessary to accommodate the wireless devices. While there are attractive reasons for using wireless, the reliability and network access leakage of wireless data, along with potential radio frequency interference, or RFI, and electromagnetic interference, or EMI, can complicate wireless use. Note that EPRI is currently investigating use of Long Term Evolution, or LTE, technology for use in a nuclear power plant. The 300 – 1,000 MHz band that LTE operates within appears to be a credible wireless solution, particularly for monitoring applications.

Quality of Service / Fault Tolerance

Resiliency is crucial for the success of the ICS using HTTPN. Maintaining high quality of service, or QoS, and ensuring proper redundancy is necessary for both of these missions. Faults within network equipment will happen and when they do, proper function of the ICS must continue. This can be achieved by using

similar tactics seen in larger IT networks. For example, “dual-homing” devices with multiple network interfaces yields reliability when the primary goes down. This is especially necessary for safety functions and critical message types.

Ensuring QoS can still be a problem for some IT networks. Proper load balancing of traffic, failure domain management and efficient routing of traffic are all necessary considerations to address when creating such an environment.

IT Infrastructure

A lot of the problems noted above have solutions. Moreover, making an architecture shift to an IP-based environment provides the ability to take advantage of solutions already created for an IP-based world. For example, managed switches allow flexibility in routing and even priority for specific traffic. Network Intrusion Detection Systems, or NIDS, and network sensors allow insight into possible incoming attacks and anomalous behavior. Web Application Firewalls, or WAFs, allow operators to do parameter/value checking within an ICS – something that’s difficult to do for many protocols. Furthermore, administration/management of devices is much easier to maintain, as each device is accessible. This suggests the ability to do auditing, testing, maintenance and even software/firmware updates easily and remotely (or at least from a defined control room). In short, there are a lot of available technologies to take advantage of in an IP-based network.



Section 6: Feasibility

Quite possibly the biggest hurdle in this push to a more secure architecture for ICS networks is on the device support side. Using the most advanced cryptography necessitates adequate computation power. In order to provide sufficient availability, with no system delay, the overhead of crypto must be met by an increase in computation power. Currently, most of the ICS components are not equipped to do such.

These timing requirements need to be emphasized and detailed in a further specification. A specification could require a device to meet certain performance benchmarks, or having specific system timing requirements.

The same should be stressed about networking interfaces. Typically, these devices support low-level serial bus interfaces. In order to handle HTTPN, these devices would need to handle the entire IP stack from Ethernet (or wireless) to HTTP. Again, an HTTPN specification should outline these specifics. Changes to HTTP and TLS should not affect any of the layers below it, which makes it adaptable for any device that can already run an IP stack with merely a software or firmware change.

This rough HTTPN description has explored the system requirements needed for deployment and implementation. Those requirements, with vendor support, are definitely achievable, with the expected shift from traditional analog devices to newer “smart” devices.

Also, an HTTPN-exclusive system could still be as susceptible as an HTTPS environment. Revisions of TLS/SSL have taught us that the best cryptographic methods today could be obsolete tomorrow. Therefore, it is crucial for the network engineers to update all devices when a newer, stronger cryptographic suite is available.

It is imperative, however, to *thoroughly* consider what’s been discussed and the technical challenges needed to overcome. HTTP was created as an insecure, request-response protocol; two large problems for the intended application in an ICS environment. This document has discussed technologies that serve as fixes to those problems, however, they may ultimately be the biggest restrictions. Nevertheless, in theory, this implementation could work with enough support.

Industry is driving product lines to be “smart”, allowing devices to connect to a network, building an ecosystem known as the Internet of Things (on the consumer front) and a parallel ecosystem in industry called the Industrial

Internet. Having an inherently secure singular specification that all of these devices can use would simplify design for future ICS landscapes; HTTPN could be a solution for this problem. Investigation into other protocols, like DNP3, as well as development of a new protocol that is inherently secure, should also be explored as possible solutions.

In summary, the challenges this kind of deployment would face are:

- Vendor support in designing crypto-capable devices
- Vendor support in designing full IP-stack devices for bi-directional communications
- Overcoming/mitigating overhead associated with encryption/decryption
- Vendor support for device-based authentication (for automated authentication)
- Key management – this is not unique to this environment; it simply remains a challenge for any environment that requires high integrity

HTTPN is more of a configuration change to HTTPS than a technical advancement. Many of the Request for Comments, or RFCs, mentioned in this document are from the IETF, however, they are not the only standards body that can make standards. For example, DNP3, was standardized by the International Electrotechnical Commission, or IEC. These groups, however, focus on the more technical side.

An alternative to an RFC from IETF (or the like), is publishing a standard that specifies exactly how HTTPN should be configured. This appears to be more appropriate for HTTPN. The target standard would be analogous to what the Payment Card Industry has created in their Data Security Standard, or PCI-DSS for handling credit card information. Their standard contains no technical advancements in the field, but merely rules on how data should be handled, proper architecture, and sufficient data validation thresholds.



Section 7: Gap Analysis

In this section, protocol differences between current HTTPS specifications and what an HTTPN specification might look like are identified. This discussion is not exhaustive, but should serve as a good representation for what is intended to be accomplished in the HTTPN specification. The purpose of these examples is to show how to take out the ambiguity in the specification.

HTTP Protocol Examples

- Section 8.1 of RFC 2616:

Section 8.1 discusses future versions of HTTP having more capability as it pertains to persistent connections, and there is still room for growth. Adoption of technologies that exist like those discussed in Section 3 (of this paper), for example, extend persistence for bi-directional communication – a necessity for control systems. Additionally, correct implementation of this could reduce some overhead.

- Sections 9.2, 9.4, 9.7, 9.8, 9.9 of RFC 2616:

These sections involve the OPTIONS, HEAD, PUT, DELETE, TRACE, CONNECT methods for HTTP. At this point in time, for an ICS application, there appears no necessity for those methods. As mentioned earlier, only GET and POST should be supported in HTTPN.

- Section 14 of RFC 2616:

Section 14 describes the Header Field Definitions semantics for HTTP methods. Because the intended communication for an ICS is likely not file based, there are really only a few suggested file types that HTTPN should use. For example, text/plain and text/html should be used for transferring basic text or webpages.

TLS Protocol Examples

- Section 7.4.4 of RFC 5246 (TLS Protocol) reads:

A non-anonymous server can optionally request a certificate from the client, if appropriate for the selected cipher suite. This message, if sent, will immediately follow the ServerKeyExchange message (if it is sent; otherwise, this message follows the server's Certificate message).

As discussed above and shown in Figure 3-1, HTTPN should not make this optional, but require the client to authenticate itself to the server. Verbiage for an HTTPN specification would read along the lines of:

Immediately following the ClientKeyExchange message, the client will provide the server with its certificate for the same cipher suite as the server provided. Verification of the client's certificate on the client's side is done prior to any data communication.

- Section 9 of RFC 5246 (TLS Protocol) reads:

In the absence of an application profile standard specifying otherwise, a TLS-compliant application MUST implement the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA (see Appendix A.5 for the definition).

Because HTTPN is disallowing any TLS fallback, there will just one cipher, which is the one specified in Table 3-2 above. With that said, HTTPN is aggressively forward-compatible; that is to say that when a stronger cipher is available, all devices should move to that cipher. It's imperative the RFC for HTTPN reflect that.

HTTP/TLS Hardening & Best Practices

Because HTTP was created without security in mind, there have been adjustments that strengthen the security of application, both configuration on the server side as well as in-message protections. There exist plenty of public resources available for hardening an HTTP server and browsers to maintain the utmost security and privacy. The following are example suggestions that should be examined for exclusion and/or modification for an HTTPN RFC.

Security Headers

In a response served from a server, headers are included to tell the client browser how to make sense of the response. Enforcement of these headers should be policy-driven; however, compliance can be checked on both sides. Some of those header options deal with security. Some suggested security header configurations include:

- strict-transport-security – this header forces an HTTP connection to upgrade to HTTPS. This should be the de facto standard for HTTPN, and all unsecure connection attempts rejected.
- content-security-policy – this header specifies what resources to accept from which origin sites. In an environment where HTTPN is used, content should only be served from the server it is communicating with, so a policy restricting the default source to itself should be mandatory.
- x-xss-protections – this header enables the cross-site scripting filter and should always be enabled.

Server Configuration

There are also specific settings that can be edited in the server configuration. These options are not always seen from the perspective of a browser, like header responses. This configuration information is usually stored as a file on the server, like `apache2.conf`, for example, for Apache servers. Enforcement of proper configuration can be done by host-based integrity checks, as well as rejection (and possibly reporting) on the client side.

- Disabling deprecated versions of SSL/TLS (ex. `SSLProtocol -ALL +TLSv2`)
- Accepting specific IP addresses for connections – this could be used to again strengthen and enforce communication policies between devices. The adaption for extending this to also include MAC addresses would be greatly helpful
- Max clients – another layer on the server side to prevent any unapproved devices from connecting to the server
- Logging – doesn't directly help in security, however, can be extremely valuable for incident response, auditing, debugging, etc.
- Other traditional security hygiene techniques (ex. running the HTTP server as a non-privileged user, segregating web resources)

Section 8: Roadmap and Future work

Getting to a state in which HTTPN would be widely deployable requires a lot of intermediate work. There are still many questions that need to be answered before any impactful design decisions are made. This section details each step in the progression of development, however it is not fully exhaustive. The following items should be addressed in the order they are presented.

1. Timing Requirements – because of the overhead necessary to run a full IP stack and handle the best encryption, specific timing requirements must be defined. These requirements can define specifically how much total time (transmission time + propagation delay + computation time) is acceptable for important devices.

Timing is also dependent on use case. Timing requirements could be broken down into tiers of criticality as some devices have more important tasks. Control data output, for example, would carry higher importance and be more time sensitive than an auditing monitor. Safety systems, obviously, would require critical timing.

In general, without these specified, it's difficult to measure the efficiency and feasibility of using HTTP as a control protocol. This will also help drive the specifications needed for device requirements.

2. Communication characterization – defining communication attributes (ex. how often changes are made, average bytes of data sent). This relates to the timing requirements, but should be more specific as far as details with respect to the amount of data needed.

Tasks 1 and 2 should be done in unison, and would likely require data sets (and potentially site visits) from active digital power plants or alternate industrial control systems that utilize IP-based communications. Deployed devices should be cataloged and examined for feasibility testing of modern cryptography. It is difficult to estimate a realistic timeframe in which these tasks could be completed in, as control systems vary in these requirements. The more data collection, the better these systems can be modeled. At a minimum, 6 months of data acquisition (or roughly 12 distinct ICS site visits) and 6 months of analysis would be getting close to accurately characterizing the communication.

3. Investigation into other protocols – this document describes the feasibility of using HTTP as a secure control protocol. Other protocols, such as DNP3, should be investigated for ensuring that HTTP is best for the application use. This is driven by Tasks 1 and 2 and would estimate about 6 months to completely evaluate. It's important to remember at this task, moving forward,

that a diverse group of subject matter experts should be involved to engineer the most robust approach.

4. Investigation into protocol security technologies – there exists other secure communication technologies that have been introduced that operate at different levels of the communication stack. For example, IPSec [9] is a security protocol that provides secure communication, but on the network layer of the network stack. Likewise, tcpencrypt [10] is a protocol in draft, that provides encryption at the transport level. Using a technology like these would allow any application at high levels, and not be bound to HTTP overhead and actions.

This task should be examined for 6 to 12 months, depending on if physical implementation of a test network is needed; it is suggested on a small scale.

5. Initial Specification Draft – At this point, the framework of how HTTPN would work would be set, and concrete details can begin to be drafted. Deciding on where the specification will be submitted is also necessary at this stage. Targeting the correct audience would be vital here to continue the advancement of HTTPN. A fair estimate for an RFC for HTTPN would be about 12 months, as it largely uses other RFCs.
6. Request for Comments and Review –At some point in this process, workgroups comprised of industry experts from various organizations would begin to detail specification nuances, and begin to challenge it. Review and rewrite iterations could take anywhere from 12 months to 5 years, depending on the activity. *Note: This is a rough estimate, there is no set schedule for standardizing.*

A visual representation of these steps is showing the following Gantt chart, Figure 8-1.

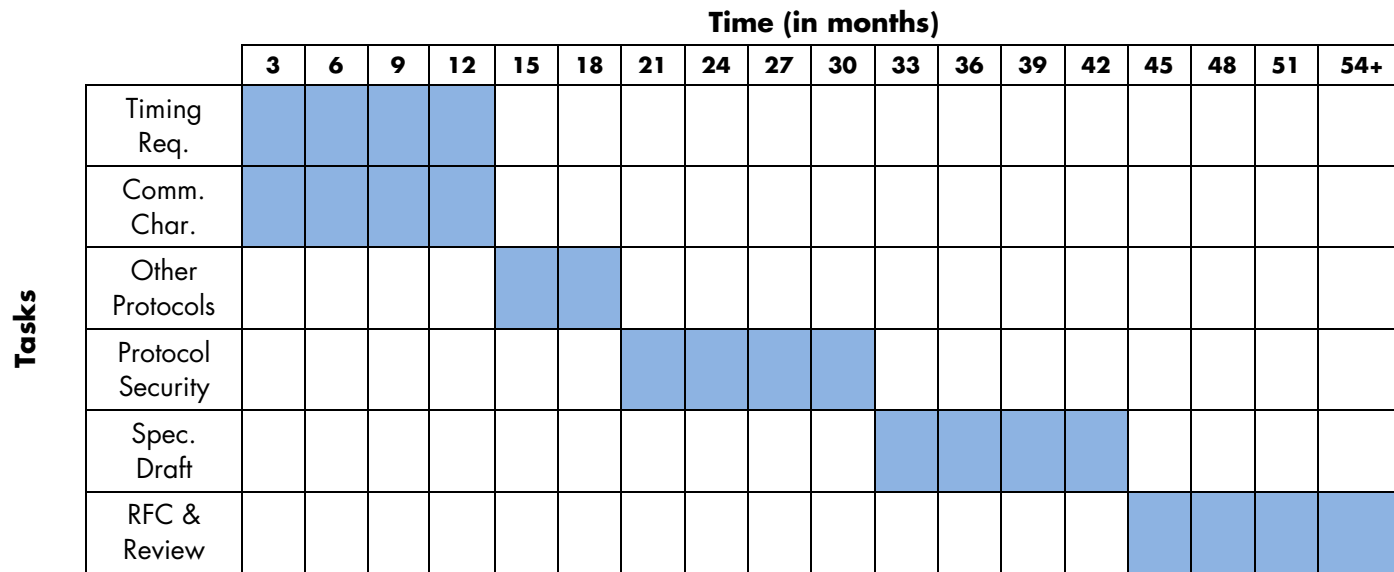


Figure 8-1
Estimated Timeline of Future Work

Section 9: References

1. Internet Engineering Task Force. *Hypertext Transfer Protocol – HTTP 1.1*, 1997. RFC 2068. <https://tools.ietf.org/html/rfc2068> [IETF RFC]
2. Internet Engineering Task Force. *Hypertext Transfer Protocol – HTTP 1.1*, 1999. RFC 2616. <https://tools.ietf.org/html/rfc2616> [IETF RFC]
3. Internet Engineering Task Force. *Hypertext Transfer Protocol Version 2 (HTTP/2)*, 2015. RFC 7540. <https://tools.ietf.org/html/rfc7540> [IETF RFC]
4. Internet Engineering Task Force. *HTTP over TLS*, 2000. RFC 7540. <https://tools.ietf.org/html/rfc2818> [IETF RFC]
5. Internet Engineering Task Force. *The Transport Layer Security (TLS) Protocol Version 1.2*, 2008. RFC 5246. <https://tools.ietf.org/html/rfc5246> [IETF RFC]
6. Internet Engineering Task Force. *The WebSocket Protocol*, 2011. RFC 6455. <https://tools.ietf.org/html/rfc6455> [IETF RFC]
7. Internet Engineering Task Force. *Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*, 2011. RFC 6202. <https://tools.ietf.org/html/rfc6202> [IETF RFC]
8. Guarnacci, Nino. Oracle. *Introduction to Bayeux Protocol (HTTP Publish-Subscribe)*, 2009. https://blogs.oracle.com/slc/entry/introduction_to_bayeux_protocol [Blog/Presentation]
9. Internet Engineering Task Force. *Security Architecture for the Internet Protocol*, 2005. RFC 4301. <https://tools.ietf.org/html/rfc4301> [IETF RFC]
10. Tcpcrypt Org. *Tcpcrypt – Encrypting the Internet*. <http://www.tcpcrypt.org/index.php> [Website]



Appendix A: HTTPN Experiments

Measuring TLS Overhead

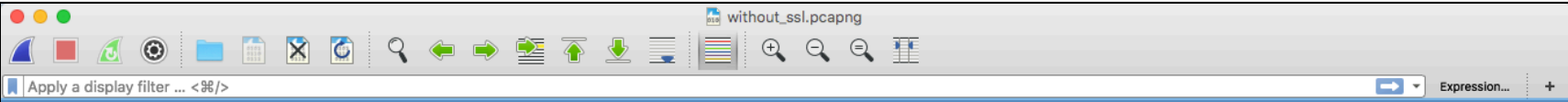
A large concern in researching HTTPN has been the latency associated with the given security. In order to realize this impact, a quick experiment looking at the latency of packets on the wire was completed. In this scenario, the default Apache2 server was installed on a virtual machine, using a base Debian OS image.

As the control, the Apache2 server was installed with no SSL/TLS, and a fetch from the host machine was done to get the home page of the server. From this, it was observed that it took approximately 6.031 milliseconds from the beginning TCP handshake to the final acknowledgement of the data sent. The last two packets show the ending acknowledgement in the transaction, likely due to timeout.

Following this, SSL/TLS was enabled on the webserver. To accurately characterize HTTPN, the server was configured with the cryptographic suite suggested in Table 3-2 of this paper, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384. A capture of this is shown in Figure A-2.

In this experiment, it was observed that the same transaction took about 27.360 milliseconds to complete, over three times the duration for the first test. This is due to the necessary communication done to configure encryption.

It should be acknowledged, however, that this overhead would not be realized on every transaction. The TLS handshake need not be done for each transaction, if the connection remains active. Overtime, the overhead would diminish to the overall burden of the transaction.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000				66	<Ignored>
2	0.000231				66	<Ignored>
3	0.001332	172.16.45.1	172.16.45.129	TCP	78	55589->80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1003235762 TSecr=...
4	0.001461	172.16.45.129	172.16.45.1	TCP	74	80->55589 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSva...
5	0.001496	172.16.45.1	172.16.45.129	TCP	66	55589->80 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=1003235762 TSecr=806404
6	0.002073	172.16.45.1	172.16.45.129	HTTP	419	GET / HTTP/1.1
7	0.002184	172.16.45.129	172.16.45.1	TCP	66	80->55589 [ACK] Seq=1 Ack=354 Win=30080 Len=0 TSval=806404 TSecr=1003235762
8	0.003130	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
9	0.003136	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
10	0.003152	172.16.45.129	172.16.45.1	HTTP	587	HTTP/1.1 200 OK (text/html)
11	0.003780	172.16.45.1	172.16.45.129	TCP	66	55589->80 [ACK] Seq=354 Ack=2897 Win=129600 Len=0 TSval=1003235763 TSecr=8...
12	0.003912	172.16.45.1	172.16.45.129	TCP	66	55589->80 [ACK] Seq=354 Ack=3418 Win=130528 Len=0 TSval=1003235763 TSecr=8...
13	0.006926	172.16.45.1	172.16.45.129	HTTP	382	GET /icons/openlogo-75.png HTTP/1.1
14	0.007207	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
15	0.007214	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
16	0.007217	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
17	0.007220	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
18	0.007222	172.16.45.129	172.16.45.1	HTTP	314	HTTP/1.1 200 OK (PNG)
19	0.007329	172.16.45.1	172.16.45.129	TCP	66	55589->80 [ACK] Seq=670 Ack=6314 Win=128160 Len=0 TSval=1003235766 TSecr=8...
20	0.007354	172.16.45.1	172.16.45.129	TCP	66	55589->80 [ACK] Seq=670 Ack=9210 Win=130816 Len=0 TSval=1003235766 TSecr=8...
21	0.007363	172.16.45.1	172.16.45.129	TCP	66	55589->80 [ACK] Seq=670 Ack=9458 Win=130560 Len=0 TSval=1003235766 TSecr=8...
22	0.018847	172.16.45.1	172.16.45.129	TCP	66	55549->80 [FIN, ACK] Seq=1 Ack=1 Win=4101 Len=0 TSval=1003235777 TSecr=801...
23	0.019006	172.16.45.129	172.16.45.1	TCP	66	80->55549 [ACK] Seq=1 Ack=2 Win=235 Len=0 TSval=806408 TSecr=1003235777
24	0.020136				78	<Ignored>

Figure A-1
GET of Homepage without SSL/TLS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000				42	<Ignored>
2	0.000220				60	<Ignored>
3	0.000233	172.16.45.1	172.16.45.129	TCP	78	55218->443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=10020709...
4	0.000300	172.16.45.129	172.16.45.1	TCP	74	443->55218 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM...
5	0.000339	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=1002070916 TSecr...
6	0.000753	172.16.45.1	172.16.45.129	TLSv1.2	319	Client Hello
7	0.000952	172.16.45.129	172.16.45.1	TCP	66	443->55218 [ACK] Seq=1 Ack=254 Win=30080 Len=0 TSval=514649 TSecr=10...
8	0.003229	172.16.45.129	172.16.45.1	TLSv1.2	1227	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9	0.003288	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=254 Ack=1162 Win=130592 Len=0 TSval=1002070919 ...
10	0.021324	172.16.45.1	172.16.45.129	TLSv1.2	141	Client Key Exchange
11	0.021350	172.16.45.1	172.16.45.129	TLSv1.2	72	Change Cipher Spec
12	0.021363	172.16.45.1	172.16.45.129	TLSv1.2	111	Encrypted Handshake Message
13	0.021649	172.16.45.129	172.16.45.1	TCP	66	443->55218 [ACK] Seq=1162 Ack=380 Win=30080 Len=0 TSval=514654 TSecr...
14	0.021961	172.16.45.129	172.16.45.1	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
15	0.021995	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=380 Ack=1213 Win=131008 Len=0 TSval=1002070936 ...
16	0.022433	172.16.45.1	172.16.45.129	TLSv1.2	418	Application Data
17	0.023391	172.16.45.129	172.16.45.1	TLSv1.2	1514	Application Data, Application Data
18	0.023398	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
19	0.023401	172.16.45.129	172.16.45.1	TLSv1.2	703	Application DataApplication Data
20	0.023442	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=732 Ack=4109 Win=128160 Len=0 TSval=1002070938 ...
21	0.023456	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=732 Ack=4746 Win=127520 Len=0 TSval=1002070938 ...
22	0.023462	172.16.45.1	172.16.45.129	TCP	66	[TCP Window Update] 55218->443 [ACK] Seq=732 Ack=4746 Win=131072 Len...
23	0.027018	172.16.45.1	172.16.45.129	TLSv1.2	412	Application Data
24	0.027494	172.16.45.129	172.16.45.1	TLSv1.2	1514	Application Data
25	0.027501	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
26	0.027504	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
27	0.027507	172.16.45.129	172.16.45.1	TCP	1514	[TCP segment of a reassembled PDU]
28	0.027510	172.16.45.129	172.16.45.1	TLSv1.2	372	Application Data
29	0.027553	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=1078 Ack=7642 Win=128160 Len=0 TSval=1002070941...
30	0.027578	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=1078 Ack=10538 Win=125280 Len=0 TSval=100207094...
31	0.027584	172.16.45.1	172.16.45.129	TCP	66	[TCP Window Update] 55218->443 [ACK] Seq=1078 Ack=10538 Win=131072 L...
32	0.027593	172.16.45.1	172.16.45.129	TCP	66	55218->443 [ACK] Seq=1078 Ack=10844 Win=130752 Len=0 TSval=100207094...
33	0.037033				78	<Ignored>
34	0.037186				74	<Ignored>

Figure A-2
GET of Homepage with SSL/TLS

Testbed Overview

During the development of the standards needed for HTTPN as a control protocol, it is important to understand the impact any design decisions will have on the control system itself. In order to better understand these effects, a platform that allows testing on a physical control system is needed.

The use of this testbed will ensure that when the HTTPN standards are finalized, they will not be found to be unusable on an actual control network due to excessive demands on the system, or due to any other concerns that may arise. For example, hardware typically found in control networks, such as Programmable Logic Controllers, or PLCs, and Remote Terminal Units or RTUs, generally do not have encryption capabilities built in. Adding these requirements will significantly increase the processing requirements of these components, which we must ensure does not slow down the network enough such that it degrades its ability to exert control over the physical system it runs. By testing the HTTPN protocol in the lab and ensuring it can be applied in a control environment without adversely affecting the performance of the system, these issues can be avoided.

Testbed Setup

This testbed consists of 4 Raspberry Pis and 2 Beaglebone Blacks, which have been connected through a Cisco SG300 switch. The 4 Pis run Ubuntu MATE and have been purposed to act as a Certificate Authority (CA), a Human Machine Interface (HMI), a Controller, and another for various functions such as a web server or historian. The 2 Beaglebones act as Remote Terminal Units (RTUs) and run the Angstrom distribution of Linux. This network is shown in Figure A-1.

Currently a simple PID control loop has been implemented for the controller. A 12-volt fan is used as the actuator for this system, while the sensor is an anemometer. The variable of interest is the speed of the anemometer. Both the actuator and sensor are operated locally by a RTU, which acts to provide input and output for the system. The necessary connections for wiring the Beaglebones to the system hardware can be seen in the figures below. A 14-volt Tripp-Lite power supply is used to provide steady power for the fans and servo motor.

Also, note that this control loop must compensate for outside disturbances, which are provided by a second fan which sits on top of a servo motor. Both the second fan and the servo are not part of the control loop itself, but are there to provide for random environmental input to the system.

Sample Experiment

A simple experiment for this testbed was conducted to show what occurs with high latency issues inside the control loop. This is done by incorporating an artificial delay in the code that slows down network communications, which could potentially be caused by taking too long to process the signals or also by attacks such as Denial-of-Service, or DOS. If this latency causes the network communication to slow down enough that this delay is higher than the delay margin of the system, it will cause instability.

Experiment Analysis

Results from the experiment were about as expected. With induced delay, instability within the system rose, putting the system out of equilibrium. The exact threshold needed to create this is unknown, as it was not trivial finding exactly when it began.

Furthermore, the most difficult process in doing this experiment is configuration of the server and different endpoints. In this case, it was representative of problems operators would face in real world deployments. Additionally, the testbed proved valuable in that it is necessary for rapid development. Testing out different configurations is also very helpful for evaluating benefits in the system. This could be useful in measuring overhead and availability.

Further experimentation could also include:

- Inclusion of real hardware (PLCs, real server boxes as CA)
- Inclusion of industry common software (HMI displays, reporting/monitoring, etc.)
- Use of different HTTP persistence techniques
- Wireless compatibility, or other possible mediums.

Network Diagrams

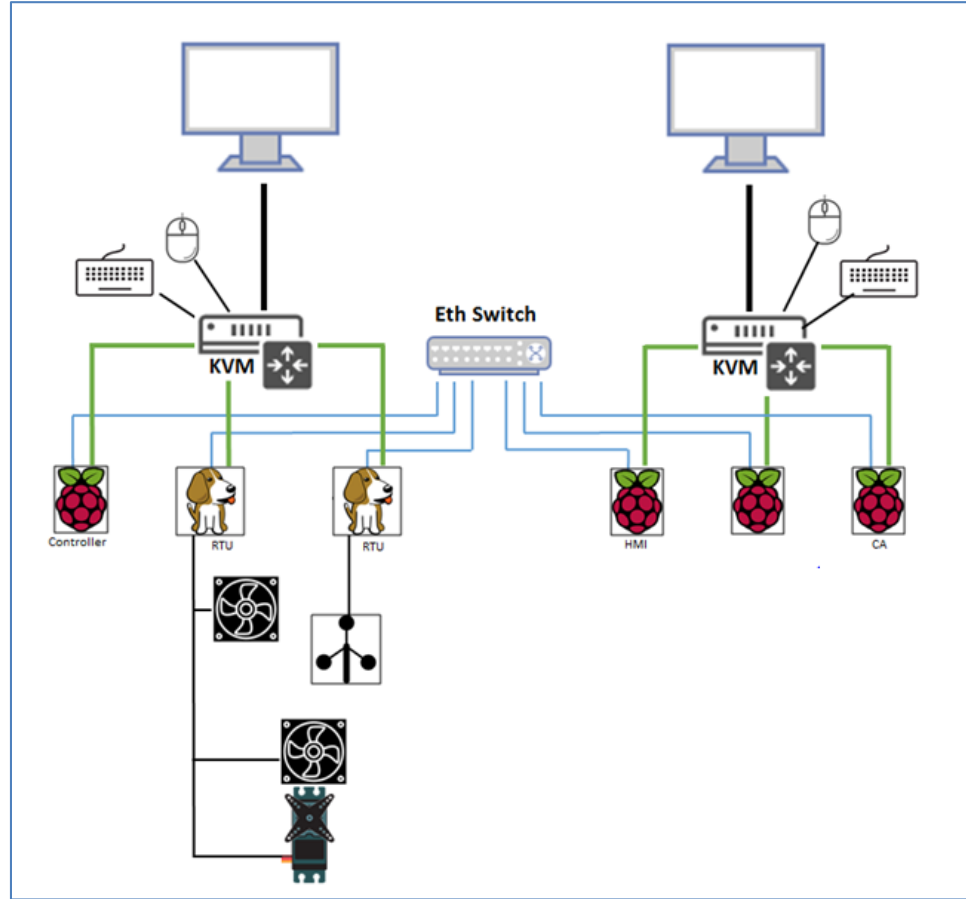


Figure A-3
Block Diagram of HTTPN Testbed

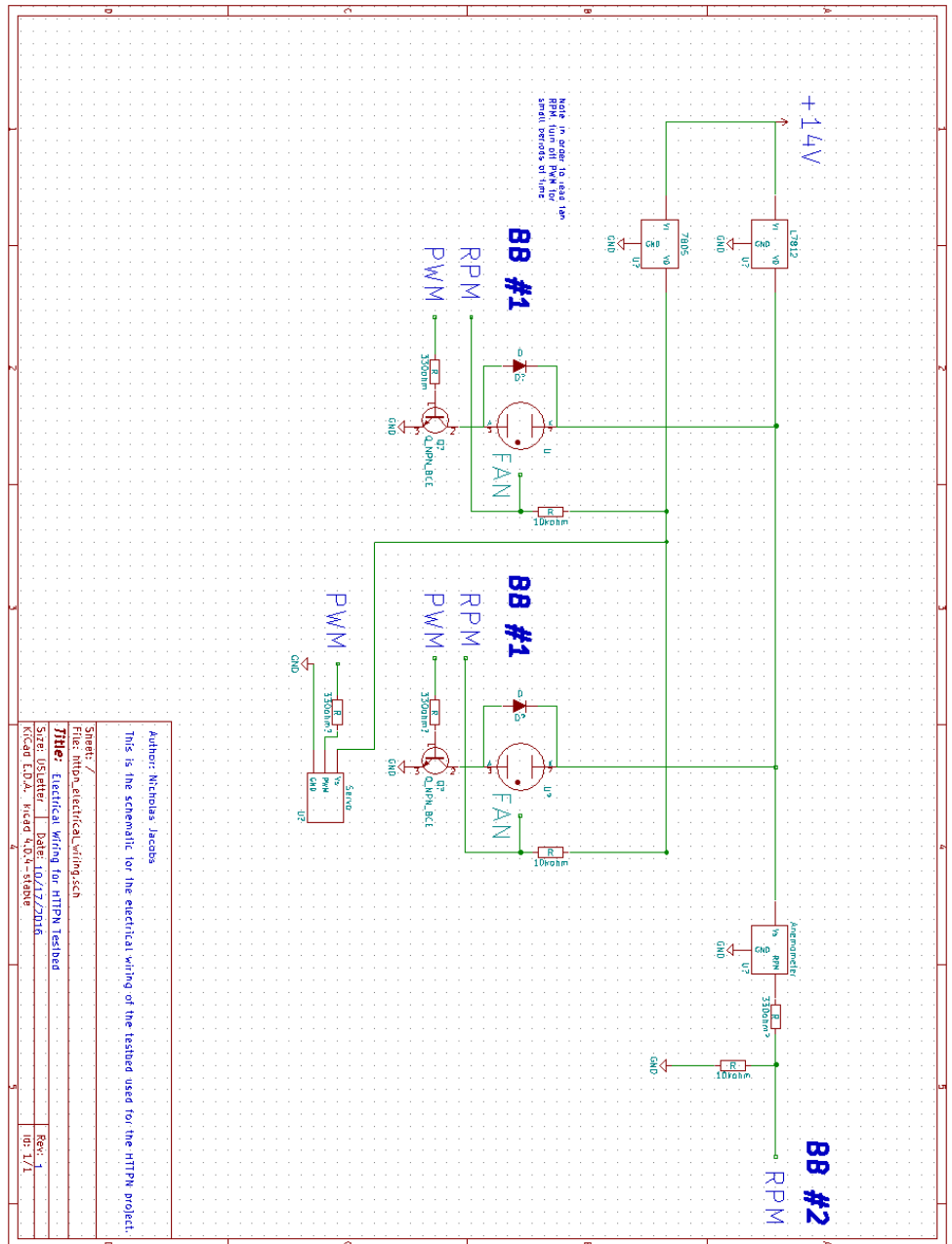


Figure A-4
HTTPN Testbed Circuit Schematic

Export Control Restrictions

Access to and use of EPRI Intellectual Property is granted with the specific understanding and requirement that responsibility for ensuring full compliance with all applicable U.S. and foreign export laws and regulations is being undertaken by you and your company. This includes an obligation to ensure that any individual receiving access hereunder who is not a U.S. citizen or permanent U.S. resident is permitted access under applicable U.S. and foreign export laws and regulations. In the event you are uncertain whether you or your company may lawfully obtain access to this EPRI Intellectual Property, you acknowledge that it is your obligation to consult with your company's legal counsel to determine whether this access is lawful. Although EPRI may make available on a case-by-case basis an informal assessment of the applicable U.S. export classification for specific EPRI Intellectual Property, you and your company acknowledge that this assessment is solely for informational purposes and not for reliance purposes. You and your company acknowledge that it is still the obligation of you and your company to make your own assessment of the applicable U.S. export classification and ensure compliance accordingly. You and your company understand and acknowledge your obligations to make a prompt report to EPRI and the appropriate authorities regarding any access to or use of EPRI Intellectual Property hereunder that may be in violation of applicable U.S. or foreign export laws or regulations.

The Electric Power Research Institute, Inc. (EPRI, www.epri.com) conducts research and development relating to the generation, delivery and use of electricity for the benefit of the public. An independent, nonprofit organization, EPRI brings together its scientists and engineers as well as experts from academia and industry to help address challenges in electricity, including reliability, efficiency, affordability, health, safety and the environment. EPRI members represent 90% of the electric utility revenue in the United States with international participation in 35 countries. EPRI's principal offices and laboratories are located in Palo Alto, Calif.; Charlotte, N.C.; Knoxville, Tenn.; and Lenox, Mass.

Together...Shaping the Future of Electricity

Program:

Technology Innovation

© 2016 Electric Power Research Institute (EPRI), Inc. All rights reserved. Electric Power Research Institute, EPRI, and TOGETHER...SHAPING THE FUTURE OF ELECTRICITY are registered service marks of the Electric Power Research Institute, Inc.

3002008039

Electric Power Research Institute

3420 Hillview Avenue, Palo Alto, California 94304-1338 • PO Box 10412, Palo Alto, California 94303-0813 USA
800.313.3774 • 650.855.2121 • askepri@epri.com • www.epri.com