NAFEM®
North American Association of
Food Equipment Manufacturers

NAFEM DATA PROTOCOL

System and Data Security Best Practices

PURPOSE OF THIS DOCUMENT

This document describes a typical equipment interface security scheme, and the reasoning behind it. This document can be used as a starting point for a robust system design, but due to the evolving nature of digital security issues, this document is not an exact guide for implementing systems-level digital security. NAFEM assumes no liability for applications or the design of specific implementations based upon this document. Users of this document are responsible for their own design.

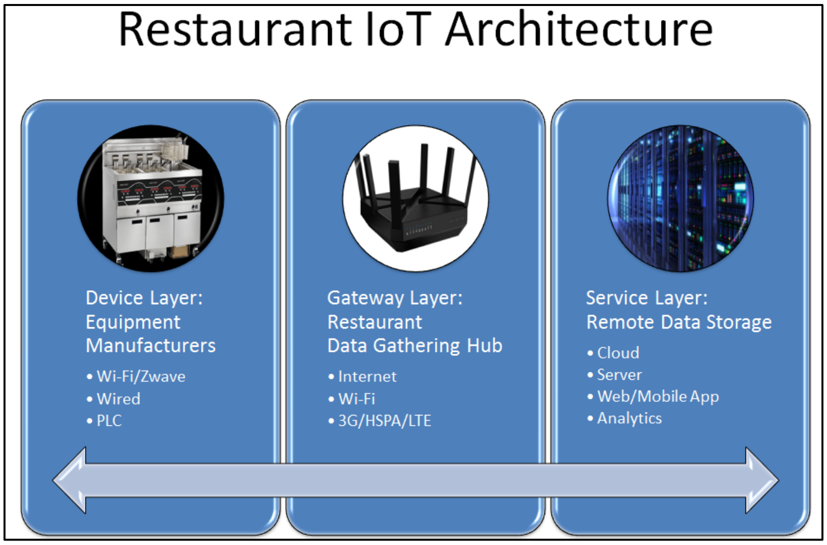
WHY NOT JUST RELY ON BUILT-IN ENCRYPTION?

A modern restaurant equipment communication network can be described as an “Internet of Things” (IoT) implementation. These IoT systems are usually designed to run autonomously and aggregate data at a few key points. The Local Gateway and Server/Cloud are the primary collection nodes for restaurant equipment data networks.

This document can be applied to any equipment that is capable of communication, whether it is through the Internet or local wired or wireless communication protocols.

Restaurant IoT data will pass through both private and public networks on its journey to the Server/Cloud/Data Lake. Wireless data encryption within the restaurant is a start, but there is a long history of Gateway, Router, and Server data breaches. Since equipment data is decrypted and re-encrypted at each step of the transport process, this paper explains the need for end-to-end (or application layer) data security.

This document focuses on complete application layer security, and only relies on transport-level security (such as Wi-Fi WPA with PSK or HTTPS encryption) as an additional measure in an overall secure system.



GLOSSARY OF TERMS

AES: Advanced Encryption Standard is a symmetric block cipher chosen by the U.S. government to protect classified information. There are many variants with periodic updates responding to various attacks and improvements in computing power.

Gateway: A device that enables machine-to-machine communication by connecting appliances in the home, workplace or smart city to cellular, Wi-Fi, or public internet. Gateways are built on chipsets that include low-power connectivity (typically ZigBee, Bluetooth, or 900Mhz) and data processing capabilities.

IP Connected, IP Addressable, Internet Connected: Any device or equipment that is connected to the internet. This includes wired and wireless internet connectivity. Effort should be made to minimize the visibility of devices to the Internet in general, and have a single gateway per physical location as it minimizes ongoing security costs and risks.

NIST: National Institute of Standards and Technology.

RSA: Public-key cryptosystems widely used for secure data transmission. Named after the inventor’s initials: Ron Rivest, Adi Shamir, and Leonard Adleman.

Server: A server is a computer that provides data to other computers. It may serve data to systems on a local area network (LAN) or a wide area network (WAN) over the Internet. While server software is specific to the type of server, the hardware is not as important. A regular desktop computer can be turned into a server by adding the appropriate software.

SHA: Secure Hash Algorithm.

SECURITY MODEL FEATURES

A typical Data Security Model will address the following three key aspects: Confidentiality, Integrity, and Authentication.

Confidentiality is the “property that information is not made available or disclosed to unauthorized individuals, entities, or processes" (ISO/IEC 27000:2016). Using AES, RSA, or other approved methods, data packets are encrypted, ensuring confidentiality throughout the transport process.

Integrity is the property that ensures data has not changed between the point of origin and point of destination. Encryption methods usually make no inherent provisions for data integrity, and the burden of ensuring data integrity falls upon the application-level data protocol.

Authentication is the property of all components of a data system to prove “they are who they say they are”. This is where IoT applications present the greatest need for innovative solutions, as authentication was typically performed by a user inputting a password before using a service. IoT devices are meant to run autonomously, and a different authentication scheme must be used.

Additionally, it is recommended to avoid a single key/password per device model, as that concept caused immeasurable damages during the first decade of home internet usage. Learning the key/password for a single unit of a popular model of Wi-Fi router or modem gave direct access to millions of PCs.

IoT Authentication can be better implemented by having a field configurable scheme where the field technician “programs” the key or certificate into the device during installation. This process could entail using automatic configuration software on the Gateway, or having a dedicated piece of software or hardware that sets the keys during initial deployment and site configuration.

ACHIEVING DATA CONFIDENTIALITY WITH ENCRYPTION

There are two major types of encryption traditionally used for transmitting data. Both types of encryption have advantages and limitations; however, neither one is inherently better overall:

- **Symmetric Key Encryption**: This method of encryption uses a single key to encrypt and decrypt the data. The device and the server use the same key for all communications.

  Major advantages include:

  - Higher performance and more flexible for various packet data sizes.
  - Simple key management scheme for device->server communications.
  - Lower engineering cost to implement.

  Major challenges include:

  - Need to send the key during configuration, which has some vulnerability.
  - Difficult to manage machine to machine communications due to need of having every device’s key stored locally.

- **Asymmetric (or Public/Private Key) Encryption**: This method uses a widely held public key to read incoming messages, and keeps a single copy of the private key to create a message.

  Major advantages include:

- Most secure way to register devices and equipment with less attack vectors.
- Improved ability to perform machine to machine communications.

Major challenges include:

- Higher processing power required to create and read messages.
- Higher engineering costs to implement.

To create the most effective encryption scheme, engineers will use a combination of both encryption methods: Asymmetric to register a device/key, then Symmetric to perform the communications itself. That method leverages the best of both methods, but is often not practical for smaller embedded systems or lower volume production runs. Therefore, using only symmetric key encryption (AES-256) and having a relatively safe device registration process is more widely used. The key registration can either be done via factory programming, or by using a “factory key” to send the initial device setup information to the server.

Either method can be made to work; it is a matter of choice and scaling considerations that decide whether to go into the asymmetric key algorithms for device registration, or using a secure factory or wired programming approach.

ENCRYPTION EXAMPLE

Example from section 11.0 of NDPv3 App Note - Equipment Identification Request Message:

CMD_PKT0123øøøø01000

FIGURE 1 - COMMAND PACKET

‚$ù'_o©L¡ß»«¸µ*w□h8Rçä□ P}ëâþö

FIGURE 2 – AES-256 ENCRYPTED CMD PACKET

ENCRYPTION RESOURCES

To help find a suitable encryption method, the National Institute of Standards and Technology has implemented a certification listing website, most major software providers are represented.

NIST AES Algorithm Validation List: FIPS-197 Symmetric Algorithms
http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html

NIST RSA Algorithm Validation List: FIPS-186 Asymmetric Algorithms
http://csrc.nist.gov/groups/STM/cavp/documents/dss/rsanewval.html

OpenSSL - Open Source Encryption Library: Software package suitable for implementing equipment encryption. Algorithms and code for both Symmetric (AES) and Asymmetric (RSA) encryption: NIST listed for both AES and RSA
https://www.openssl.org

ACHIEVING DATA INTEGRITY

To ensure a message has not been modified in transit, a summary field is usually appended to the message. This summary field or signature is then encrypted with the original message, and allows the receiver to perform a validation test. If the receiver is unable to match the supplied summary code through performing an agreed-upon operation, the message is discarded.

As with encryption, the engineer has a choice when deciding on how to achieve data integrity:

Cyclic Redundancy Check (CRC): Provides a computationally efficient method to detect for data errors. CRCs do not provide protection against malicious packet changes. The very small size of the CRC does not add significant overhead to the communication protocol.

Hashing Algorithm: Hashes are a more complex algorithm that can be used to verify the authenticity of a device or message, as well as prevent unintentional errors during transmission. The resultant hash code to be appended is much larger, and approaches the size of most typical IoT data packets.

Example from section 11.0 of NDPv3 App Note – Comparing CRC-32 vs SHA-256 lengths

```
EQUIP_IDFryerøøøøøøøøøøøøøøøøøøøøøøøøøøøA1FryerCoø1234567
890øøøøøøøøøøøøøøøøøøøøV3.00.05øøøøøøø
```

FIGURE 3 - ORIGINAL EQUIPMENT IDENTIFICATION MESSAGE

```
51187176b3b33af9c7ad771091653c1744191df4bf0f4c61909c15c7bbc8e845
```

FIGURE 4 - SHA-256 HASH OUTPUT FOR EQUIPMENT IDENTIFICATION MESSAGE

```
TDrY
```

FIGURE 5 - CRC-32 FOR EQUIP ID MESSSAGE

Either of these outputs (figure 4 or 5) would be **appended the original message**, before encryption, to allow the receiver to perform the same calculation and compare the result.

DATA INTEGRITY RESOURCES

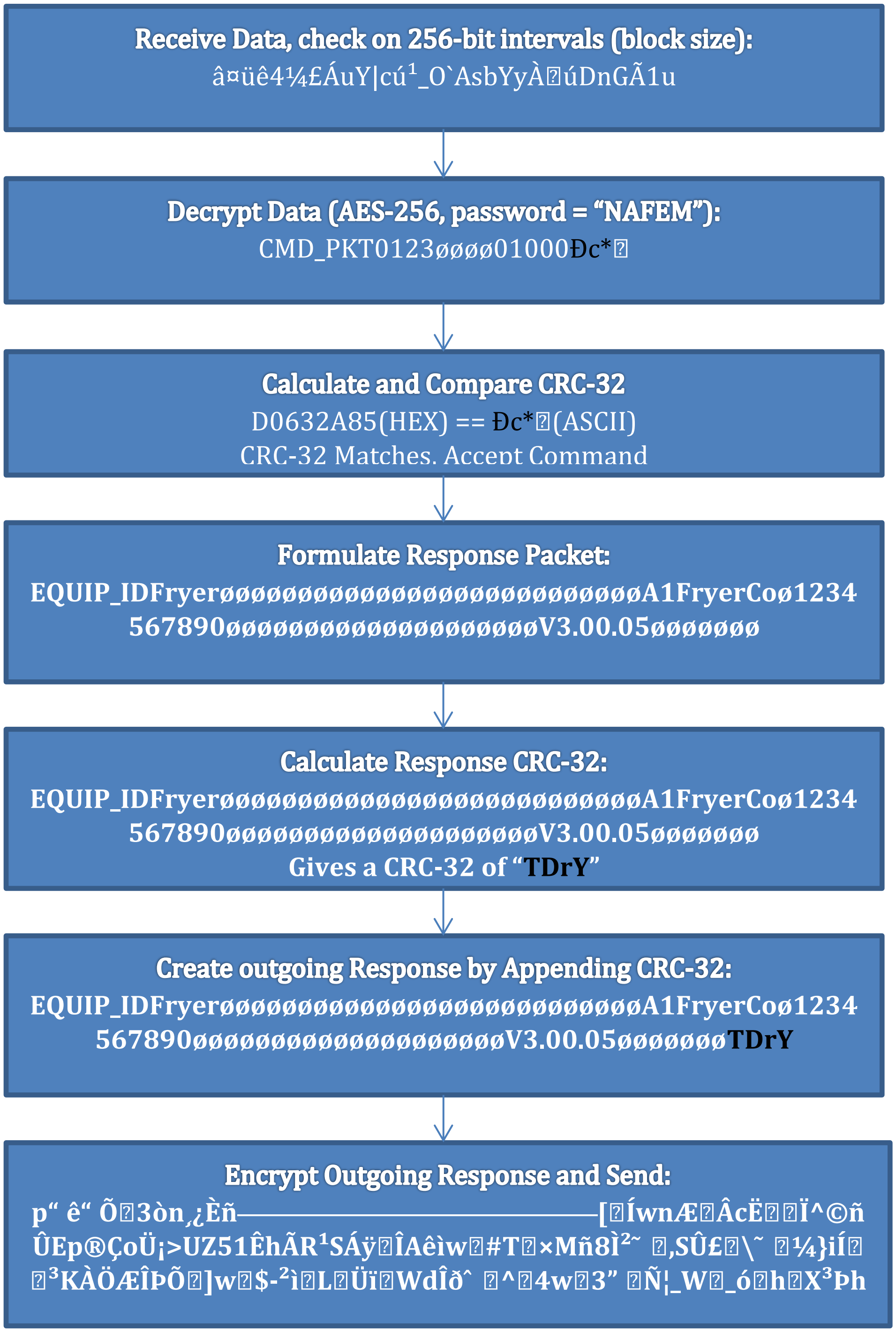
If using a CRC, there are no governing bodies to approve the algorithm. Finding a CRC algorithm is generally supported by most hardware and software providers directly.

If using a hashing function like SHA-256, NIST has a similar program to encryption, as a hash is used for higher security applications and is subject to attacks due to its capabilities. OpenSSL has many hashing algorithms that are NIST approved.

NIST Secure Hash Algorithm Validation List: FIPS 180-3 Secure Hash Standard
http://csrc.nist.gov/groups/STM/cavp/documents/shs/shaval.htm

PUTTING IT ALL TOGETHER

To show an example of a minimum viable data protection scheme, a combination of integrity and security measures are used to ensure proper system operation. The following example will walk through a typical transaction and show how the data transforms at each step within an example piece of equipment.



Receive Data, check on 256-bit intervals (block size):
â¤üê4¼£ÁuY|cú¹_O`AsbYyÀ□úDnGÃ1u

Decrypt Data (AES-256, password = "NAFEM"):
CMD_PKT0123øøøø01000Ðc*□

Calculate and Compare CRC-32
D0632A85(HEX) == Ðc*□(ASCII)
CRC-32 Matches. Accept Command

Formulate Response Packet:
EQUIP_IDFryerøøøøøøøøøøøøøøøøøøøøøøøøøøøA1FryerCoø1234
567890øøøøøøøøøøøøøøøøøøøøV3.00.05øøøøøøø

Calculate Response CRC-32:
EQUIP_IDFryerøøøøøøøøøøøøøøøøøøøøøøøøøøøA1FryerCoø1234
567890øøøøøøøøøøøøøøøøøøøøV3.00.05øøøøøøø
Gives a CRC-32 of "TDrY"

Create outgoing Response by Appending CRC-32:
EQUIP_IDFryerøøøøøøøøøøøøøøøøøøøøøøøøøøøA1FryerCoø1234
567890øøøøøøøøøøøøøøøøøøøøV3.00.05øøøøøøøTDrY

Encrypt Outgoing Response and Send:
p" ê" Õ□3òn¸¿Èñ————————————————[□ÍwnÆ□ÂcË□□Ï^©ñ
ÛEp®ÇoÜ¡>UZ51ÊhÃR¹SÁÿ□ÎAêìw□#T□×Mñ8Ì²˜ □,SÛ£□\˜ □¼}iÍ□
□³KÀÖÆÎÞÕ□]w□$-²ì□L□Üï□WdÎðˆ □^□4w□3" □Ñ¦_W□_ó□h□X³Þh

MANAGING THE DATA

Encryption and decryption is often easier to implement at the server, or if using a more advanced equipment control board (such as one with embedded Linux). The following resources cover some of the most widely-used programming environments for servers and high performance embedded systems, they can be used as a way to efficiently setup and test an end-to-end encryption scheme.

Java – Javax.Crypto.Cipher Class:
https://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html

.NET (C#) - System.Security.Cryptography.AesManaged Class
https://msdn.microsoft.com/en-us/library/system.security.cryptography.aesmanaged(v=vs.110).aspx

NEXT STEPS

The other major consideration when developing a security model is encryption key management. While most equipment will only need encryption to function securely; gateways, servers, and any device directly connected to the internet will be routinely subject to attacks. A typical public IP address may be attacked 100's of times per day, and that number appears to rise over time.

For key/password management, Java has many built in features to securely store encryption keys within a large system. .NET has less native support, but there are plenty of 3rd party key management solutions available for easy implementation.

In embedded systems, the keys can be stored in secure flash, but care must be taken to protect the data from being read out via JTAG and other debug interfaces. Since physical security is almost never available for embedded controls, key management and exposure should be developed to avoid ever having the keys available in a human readable or accessible format.

SUMMARY

While all the communications links in an equipment network may contain inherent security and integrity, the data will be vulnerable as it is translated between the collection and storage nodes. Implementing end-to-end application layer security can increase the security and robustness of a system. Recent widely publicized attacks using IoT devices imply that there may be a shift in liability towards manufacturers if proper precautions are not taken. While it may add complexity to device development, there are many organizations to support the development and implementation of good practices.