More on PrimeCR and advanced gateway lock communication

August 11th, 2025

ALLEGION™

PIONEERING SAFETY™

The database file

- The database file contains the cards and timezones they are allowed, as well as auto-unlocks and holidays. It can also contain configuration information. It is the most important part of lock communication.
- The exact specification is in file ENGAGE_JSON_Data_Structures - 2.18.pdf
- This file is the same whether it is communicated via Wi-Fi (210), Ethernet/IP (410IP) or Bluetooth app (200).
- Let's add the card from our last example to a database file and download to the lock.

ALLEGION

Picking up where we left off

- Before starting, I recommend reviewing the document “ENGAGE Credential Sort and Encryption”
- On our final slide in the previous presentation, we can calculated raw data from a stamped number and knowledge of site code and card format.
  - Our data matched exactly what was sent back by a lock denied access event, verifying we did so correctly. The data was:

RAW DATA **100000010101111000110001000011000101**

In hex (with three 0 bits appended at the end to align to a byte boundary) this is 0x815e310628

APPENDED RAW DATA **100000010101111000110001000011000101000**

Note “left shift” and “append a trailing 0” are synonymous – this is important later.

ALLEGION

What is primeCR?

- PrimeCR refers to a json field that encodes raw data in a manner which can be used by our Engage devices.
  - PrimeCR can be encrypted or unencypted. Unless otherwise noted, we will be dealing with unencrypted data, for clarity.
- PrimeCR fields are always 16 bytes long.
  - Typically, most PACs (partners) are only interested in the first 8 bytes
  - The trailing 8 bytes are relevant if multi factor authentication is used (for instance, card plus pin number instead of just card presentation)
- PrimeCR fields deal exclusively with credential data (no time zones, etc.)
- Pin numbers are the most common 2nd credential, but can also be things like “require two cards to be presented at the bank vault door”

ALLEGION

Let's create a primeCR field from our card data

- Our pre-shift (aka append zeroes) 37-bit card data is

  `1000000101011110001100010000011000101 or 0x102BC620C5`

- We must align to the nearest byte boundary (divisible by 8), so we perform three left shift operatons to change 37 bits to 40

  `1000000101011110001100010000011000101 becomes 1000000101011110001100010000011000101000 or 0x815E310628`

  (Side note, this is equivalent to multiplying by 8. We can discuss if you're curious)

ALLEGION

Creating a primeCR - 2

- We now have 5 bytes of primeCR compatible data.

  `0x815E310618`

  - However, all primeCR fields are 16 bytes! How to solve this?
    - Straightforward – simply append 0xFF until the data is 16 bytes long

      0x815E310628FFFFFFFFFFFFFFFFFFFF

81…28 = Actual card data, with three left bit shifts

F… = appended to fill out first 8 bytes due to "only" 37 actual credential bits. Some credentials will use all 64 bits / 8 bytes of first part of Prime CR credential record

F… = appended to complete the 16 byte data structure. If we were instead creating a "card plus pin" type of credential, this would contain meaningful data

ALLEGION

Other primeCR considerations

- Since we're just getting started, we are dealing with one credential at a time. However, in practice multiple primeCR fields are put into a single json door database file and sent to an engage device
  - Generally, the best approach is to send an entire full database list of cards, pre-sorted lowest to highest value by unencrypted primeCR fields.
    - There are a few edge cases where sending differential files, instead of the full door list, is useful - but this is very rare especially with gateways involved
    - More discussion on next slide
  - Also note, unencrypted primeCR data is never actually sent to the door
    - Sort the json entries as unencrypted, then encrypt and replace the unencrypted primeCR values, but leaving the ordering based on sorted unencrypted primeCR

ALLEGION

Encrypting our PrimeCR and program the lock

- Review “Schlage ENGAGE Credential Sort and Encryption-1.34” for exact details on the encryption.
- Meanwhile, I will use my handy “NoTourPayload” tool authored by Wilson Castillo to translate my PrimeCR to an encrypted PrimeCR ready to download:
  `0x815E310628FFFFFFFFFFFFFFFFFFFF`
- encrypts to 7a749873c4b3f54dfdd31b3f82e67e55 using a site key of 00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF
- Alternately, The online tool at aes.online-domain-tools.com can be used to generate the encrypted values – this tool was set to do CBC with an initialization vector (IV) of 0 per Schlage Engage AptiQ
- If I add the record to the doorfile and upload to the lock I should get access granted (if you were able to compute your encrypted PrimeCR, substitute that value for the one in the sample json, which is far my card).
- **Use sample file “Doorfile with one card.json” in postman and swipe!**

ALLEGION

Doorfile database file considerations

- It's generally a good idea to send the entire list of cards sorted, rather than incremental changes
  - Locks have two databases of cards
    - The main database: 5000 entries (NDEB/LEB) 2000 entries (Control)
    - Secondary database 750 (NDEB/LEB) or 500 entries (Control lock)
  - The main database gets cleared by a "deleteAll = T" and then the entire array added in a door database file gets put into the main database
  - The secondary database is populated by no-tour credentials (out of scope for this discussion), and subsequent partial update door files (where "deleteAll = F")
    - This is why it's best to send the entire database at once
    - Also keep an eye out for the database user limit reached audit: 0x06000004

More details on Gateway via Ethernet

- In general, the documentation can be useful for the entire postman collection.
  - Click on “Allegion APIs_11-6-2019->View Complete Documentation”
- Gateway can download one lock at a time per gateway, although multiple individual locks spread over multiple gateways
  - Must check the progress call, not just 200 from a door file download, before concluding the doorfile was successful.
    - GET {{gatewayIP}}/edgeDevices/{{linkId}}/dbDownloadStatus
- Can also check audits for a failed doorfile application
  - Check “Database Audits (0x0600)” section of DbmAuditEvents tab of “ENGAGE - Audits - 0.22.xlsm”
  - Also the “Lock Configuration Audits (0x0400)” section can be useful

ALLEGION

Websockets

- Websockets are a standard of communication built on top of http(s)
  - See “Schlage ENGAGE WebSockets App Note-1.06.pdf“
- This communication is similar, except the gateway is the client and initiates a call to the PACs server, rather than the PACs calling URLs on the gateways
  - This make it easier for cloud based PACs
- After the initial URL call, a bidirectional socket is established
  - PACs can still access the URLs on the gateway, but can also register for real time notification via sockets of audits
  - There are some other details, such as certificate handling, a keep-alive messaging scheme, and json structures for the “events” (audits)

ALLEGION

Responsibilities of a 410IP/Ethernet Integration

- Requirements:
  - Create a site and maintain the site key
  - Enroll devices into site and link them to gateways (either through app or programmically)
  - Create doorfiles from access levels (or similar),credentials, card formats, access schedules, auto-unlock schedules and holidays (auto-unlocks/holidays for NDEB/LEB only)
  - Send out updated doorfiles whenever there are changes in access rights or credentials
  - Poll the gateways for status & audits and display as appropriate in real time (particularly display “alarms”: aka forced & propped doors, tampers, comm loss, corrupt database)
    - With websockets, polling is not needed, as gateway can initiate communication on an audit received
  - ONR: Over the network reflash of devices (refer to ONR_IP.docx)
  - Door control commands (also called manual overrides): secure, passage(NDEB/LEB only), or momentary unlock from software.

ALLEGION