

XFS4IoT SP-Dev Workgroup

20 April 2021

- Scope of the XFS Committee
 - To define and publish the XFS specification
- Vendors are 100% responsible for the design, development and testing of XFS SPs for their system
- KAL will develop an XFS4IoT C# framework for each device class
- Workgroup members can create SPs using the framework
- Workgroup members can create applications and test tools

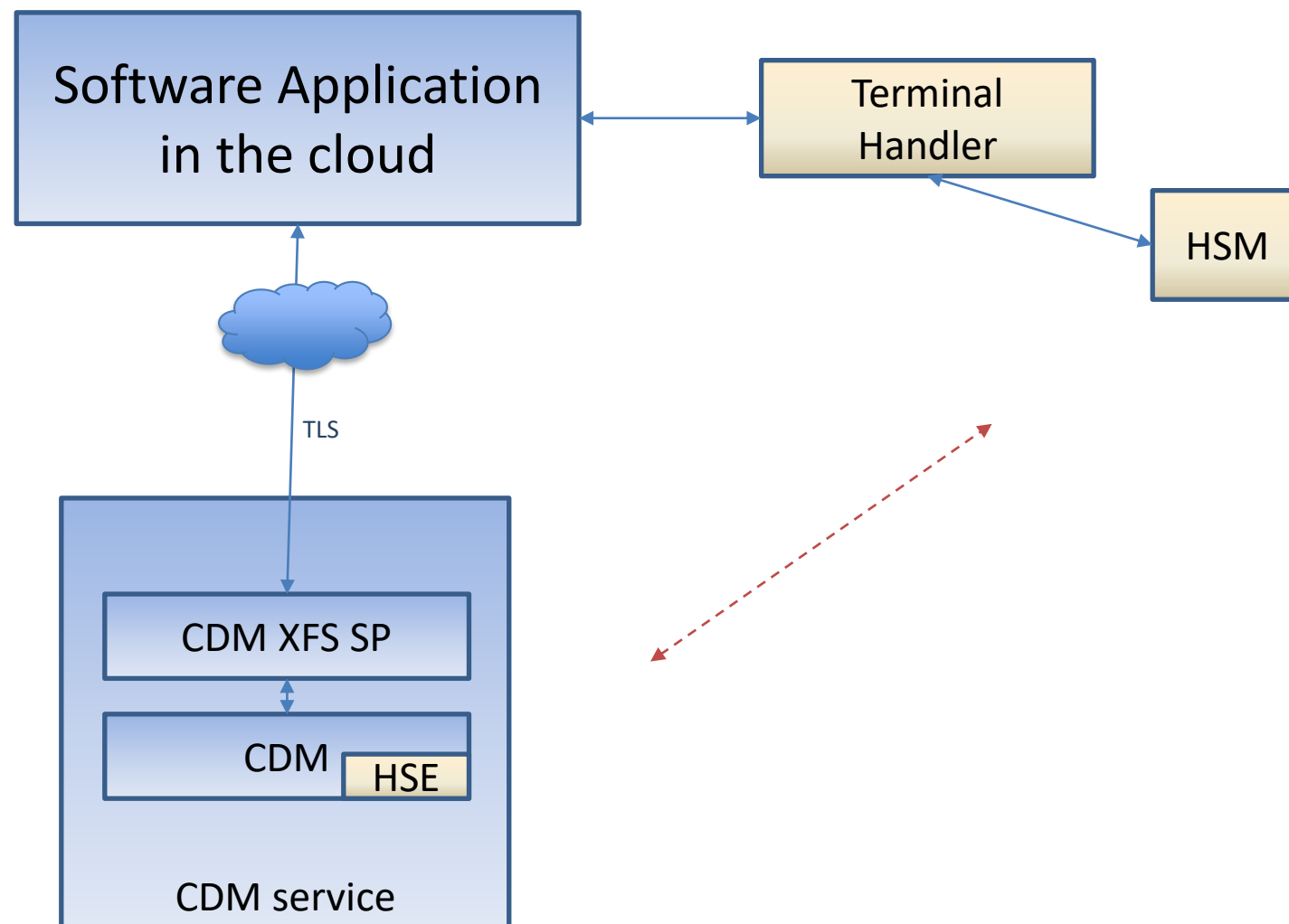
- XFS4 is not backwards compatible with XFS3 but the exchanged data is the same
- There is no XFS manager and no registry
- XFS4 uses: TLS, WebSocket, JSON messages
- We recommend that XFS3 SPs be redeveloped
- We recommend that XFS3 applications be ported to XFS4

- We looked at sample framework source code
- Framework itself does not have any hardware-specific code

- KAL is developing the framework source code
- KAL will publish framework source code as it becomes available
- The first framework will be the card reader
- We hope to have V1 of the card reader framework available on 4th May
- Members can start developing card reader SPs

- This meeting will focus on XFS4 security
 - New end-to-end security specification in XFS4IoT
- Hardware designs will need to be modified
 - XFS4IoT requires a hardware secure element (HSE) in devices such as the cash dispenser
 - Firmware needs to be able to create and verify secure tokens
 - Firmware needs to be able to store secure keys
- Applications will need to be modified
 - An HSM will be required on the application side to generate and store keys

- New end-to-end security
- Crypto capability inside CDM



Q&A and Issues in GitHub


- What commitment to the workgroup is expected?
 - There is no expectation!
- Do I have to commit to use the framework if I'm a member?
 - Again, no expectation!
- Are the XFS4IoT Specification complete?
 - No, this is a work in progress by the committee. Preview is available.


- Is the framework ready to use?
 - Same as the specification, it's a work in progress. However,...
- Will KAL also provide XFS4IoT SPs
 - Not as part of the open-source project.
- What is the timeline of XFS4IoT
 - It's up to each company to decide.


- Questions can be logged on GitHub
- Use of Issues and Q&A project

 KAL-ATM-Software / KAL_XFS4IoT_SP-Dev


 Code

 Issues 1


 Pull requests


 Actions

 Projects 1

 Wiki

 Security

 Insights

 Settings

Q&A

Updated 12 minutes ago

1 Open

+ ...

 [How to access XFS4IoT specification preview?](#) ...

#1 opened by KaderBaadoud-KAL

question

0 In Discussion

+ ...

0 Closed

+ ...



ATM Software

Security with XFS4IoT

- All the existing security from XFS3.x
 - EPP, pin entry, data encryption
 - key management
 - card reader EMV
 - etc.

- XFS4IoT is network based so requires network security
- TLS is the standard, encrypts all data on the network
- Ensures confidentiality of data on network - card reader data, printer data, etc.

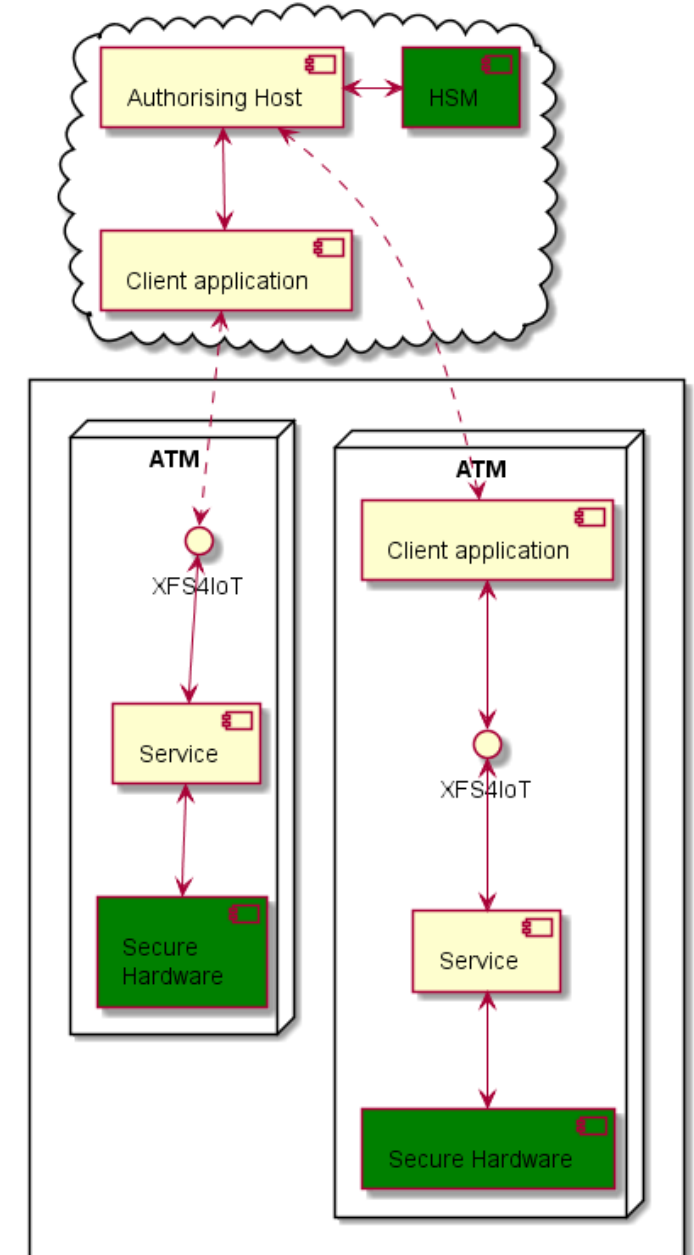
- Remote key loading with TR34
- Key exchange with TR31
- Key management can be supported by *any* device, not just
Encrypting Pin Pads (EPP) – more in a moment...

XFS 3.x by itself can have problems...

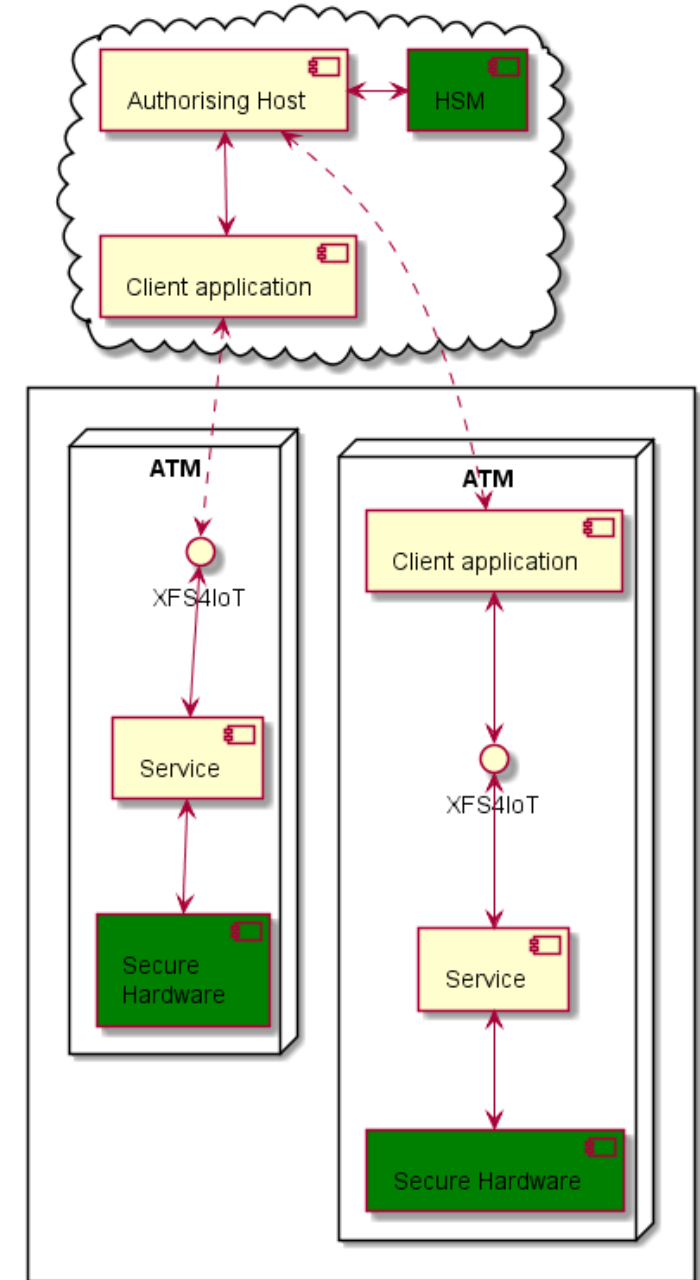


- Man in the middle attacks, black box attacks. Cash out.
- Dispenser controlled by attacker - dispenser trusts any connection
- Also network attacks - attacker changes network message to authorise invalid amount
- Need to protect authorisation
 - Make sure it comes from the real authoriser
 - Make sure it's unchanged
 - Needs to be protected all the way from the authoriser down to the hardware, across network *and* device connection

- Only end-points are trusted
 - Hardware Security Module (HSM) in cloud
 - Hardware Security Element (HSE) in device
- Transport between end points isn't trusted...



- Trust between the end points is created by sharing a secret key
- Keys are injected using remote key loading with TR34 and TR31
- Devices like dispensers will need cryptographic hardware (an HSE)



- Specification defines End-to-End security based on tokens
- A token is
 - Protected by HMAC/SHA256 – using pre-shared keys
 - Protected against replay by nonce
 - Contains all the information to validate a transaction (and no more.)
 - Very simple text - easy to deal with. Parse from firmware.

Example token



NONCE=254611E63B2531576314E86527338D61,TOKENFORMAT=1,TOKENLENGTH=0164,DISPENSE1=50.00EUR,HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F2

- NONCE=254611E63B2531576314E86527338D61,
- TOKENFORMAT=1,
- TOKENLENGTH=0164,
- DISPENSE1=50.00EUR,
- HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F

— NONCE=254611E63B2531576314E86527338D61,

Must be unique – the same value must never be used twice

Can be as simple as 1,2,3,4...

Can also be a *good* random number

Must never repeat, even after reboots

— TOKENFORMAT=1,

— TOKENLENGTH=0164,

— DISPENSE1=50.00EUR,

— HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434
916FEA6AC16F3D2F

— NONCE=254611E63B2531576314E86527338D61,

— TOKENFORMAT=1,

To support future changes to the token format

— TOKENLENGTH=0164,

Number of bytes in the token, to block adding data

Always four digits long, to make it easier to calculate

— DISPENSE1=50.00EUR,

— HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434
916FEA6AC16F3D2F

- NONCE=254611E63B2531576314E86527338D61,
- TOKENFORMAT=1,
- TOKENLENGTH=0164,
- DISPENSE1=50.00EUR,

Depends on the token type. May be multiple keys:

- Cash Dispense token - DISPENSE1
- PresentStatus token - DISPENSEID, DISPENSED1, PRESENTED1, PRESENTEDAMOUNT1, RETRACTED1

Hardware manufacturers can also add custom keys names

- HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F

- NONCE=254611E63B2531576314E86527338D61,
- TOKENFORMAT=1,
- TOKENLENGTH=0164,
- DISPENSE1=50.00EUR,
- HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434
916FEA6AC16F3D2F

Most important field

- Proves token is authentic since the key is needed to create it
- Proves the token is unchanged - change to a single bit gives a totally different HMAC

Is calculated over all other data. Always the last field.

- Token used in a command message

```
{
  "header": {
    "type": "command",
    "name": "dispenser.dispense",
    "requestId": 456
  },
  "payload": {
    ...
    "dispenseToken": "NONCE=254611E63B2531576314E86527338D61,TOKENFORMAT=1,TOKENLENGTH=0164,DISPENSE1=50.00EUR,HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F2"
    ...
  }
}
```

- Token work in both directions

```
{
  "header": {
    "requestId": 765,
    "type": "completion",
    "name": "dispenser.presentStatus"
  },
  "payload": {
    ...

    "denomination": {
      "currencies": [ { "currencyID": "EUR", "amount": 50 } ],
      ...
    },
    "presentState": "presented",
    "token" : "NONCE=1414,TOKENFORMAT=1,TOKENLENGTH=0268,DISPENSEID=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F2,DISPENSED1=50.00EUR,PRESENTED1=YES,PRESENTEDAMOUNT1=50.00EUR,RETRACTED1=NO,HMACSHA256=55D123E9EE64F0CC3D1CD4F953348B441E521BBACCD6998C6F51D645D71E6C83"
  }
}
```

- Black box attack, hardware attack - HMAC
- Man in the Middle, network attack - HMAC
- Replay attack – unique nonce in token
- Transaction Reversal Fraud (TRF) – PresentStatus token

MS Teams

Video calls every two weeks:
Tuesdays at 1300 UK time

(we will reduce to calls once a month sometime in the future)

Next call: 4th May 2021, 1300 UK, 0800 US EST, 2100 Tokyo time