mojaloop

Code Quality, Security & Compliance Improvements

PI-10 Meeting – 23 April 2020

Core Team – Kim, Godfrey, Lewis and Pedro – Crosslake
Miguel, Sam and Victor – Modusbox

Support Team – Adrian, Aime, Azeem, Creg, Donovan, Max, Miller, Philip, Renjith and Simeon

Background and Context – PI to PI View

PI 1 – P 7 (Ensure platform quality and security from design, build & release phases)

1. Coding standard
2. Signatures for non-repudiation and integrity
3. Encryption & PKI standards for preserving confidentiality
4. Established CI\CD security gates to enforce policies for :
   - Open source license usage
   - Open source vulnerabilities

PI 8 (Container security, SCA, identify & close critical gaps)

1. Introduce Container Security Gates into CI\CD
   - Docker Image Vulnerability Scanning
   - Baseline dockerfile configuration against CIS cloud security standards
2. Static Code Analysis(SQ)
3. Identify critical security gaps (Bottom up Approach)

PI 9 (Engage, standardize architecture & tackle compliance)

1. Community engagement
   - Vulnerability reporting procedure
   - Document security controls
2. Compliance Initiatives
   - Data Privacy (GDPR)
   - Plan HSM Deployment
3. Overall Security architecture review (Top Down Approach).

Key Objective

To continuously improve ***Quality and Trust*** of the Mojaloop Platform inline with industry standards.

Trust entails Reliability, Privacy, Transparency, Security, and Compliance.

Phase 1 –2 (POC) | **Phase 3 (Deployment Support)** | **Phase 4 (Go-live)**

PI-9 Objectives

1) Establish a procedure for the public to report vulnerabilities.
2) Provide a summary of all code level quality, security and compliance measures built into the Mojaloop Platform.
3) Define GDPR scope focusing on what is relevant and addressable at Mojaloop level.
4) Container security enhancements and policy customization.
5) Benchmark adopted open source tools against commercial tools.
6) Perform a platform security architecture review and standardize (Top down approach).
7) Plan HSM deployment - explore best practice design options and identify use cases.
8) Solicit Input from Implementation groups – Pain points, requests and Independent Audits.

Code Quality/Security Epic - https://github.com/mojaloop/project/issues/1213

Code Quality\Security Summary

Below are code improvement measures which have been applied on the latest code base through our CI\CD release pipeline with the key **objective** of ensuring that Mojaloop OSS Code is of ***high quality, secure and compliant*** with best practice best practice standards:

- **Open Source Quality & Vulnerability Management** – To enforce consistent use of *non-vulnerable* and *non-obsolete and non-supported* open source components and dependencies on the Mojaloop code base.

- **Open Source License Compliance** – To ensure compliance with Mojaloop open source license policy – *Apache 2.0 and the ones compatible* to it are allowed, whereas GPL / GNU aren't allowed.

- **Static Code Analysis** – To ensure that all custom code is free of well known *low coding quality issues and vulnerabilities* before any code merging, pull request or release as per our CI\CD pipeline.

- **Container and Docker Security** – To ensure compliance with our recommended *secure image configuration standards* and find\fix medium and high container based vulnerabilities in all repos.

We encourage implementation teams to adopt our open source toolset or any other toolset (open source or commercial) to address the above objectives.

Link - Project#1222 Application security summary

Vulnerability Reporting Procedure

The Objective of the procedure is to guide community members and the public on ***how security vulnerabilities should be reported safely and responsibly*** to the dedicated security team.

An overview of the vulnerability handling process:

a) The reporter reports the vulnerability privately.
b) The appropriate project's security team works privately with the reporter to resolve the vulnerability.
c) A new release of the Mojaloop package concerned is made that includes the fix.
d) The vulnerability is publicly announced to the Mojaloop Community

Migration to a bug bounty system (inclusive of a vulnerability reward programme) will be considered as we scale – FSPIOP API adoption increases and community support grows.

Link - Project#1224 Define a vulnerability reporting procedure

Mojaloop GDPR Scope

Goal :

As a "Hub Operator or Implementer", I want to "understand how the OSS community can help me fulfil my GDPR requirements", so I can "plan accordingly in my organization“

Tasks : Investigate how we can handle GDPR Requirements as a part of Mojaloop OSS Initiative.

- Produce a list GDPR controls – Completed in PI 9
- Perform a gap assessment and architecture review – deferred to PI 10
- Architecture adjustments and an implementation plan - deferred to PI 10.

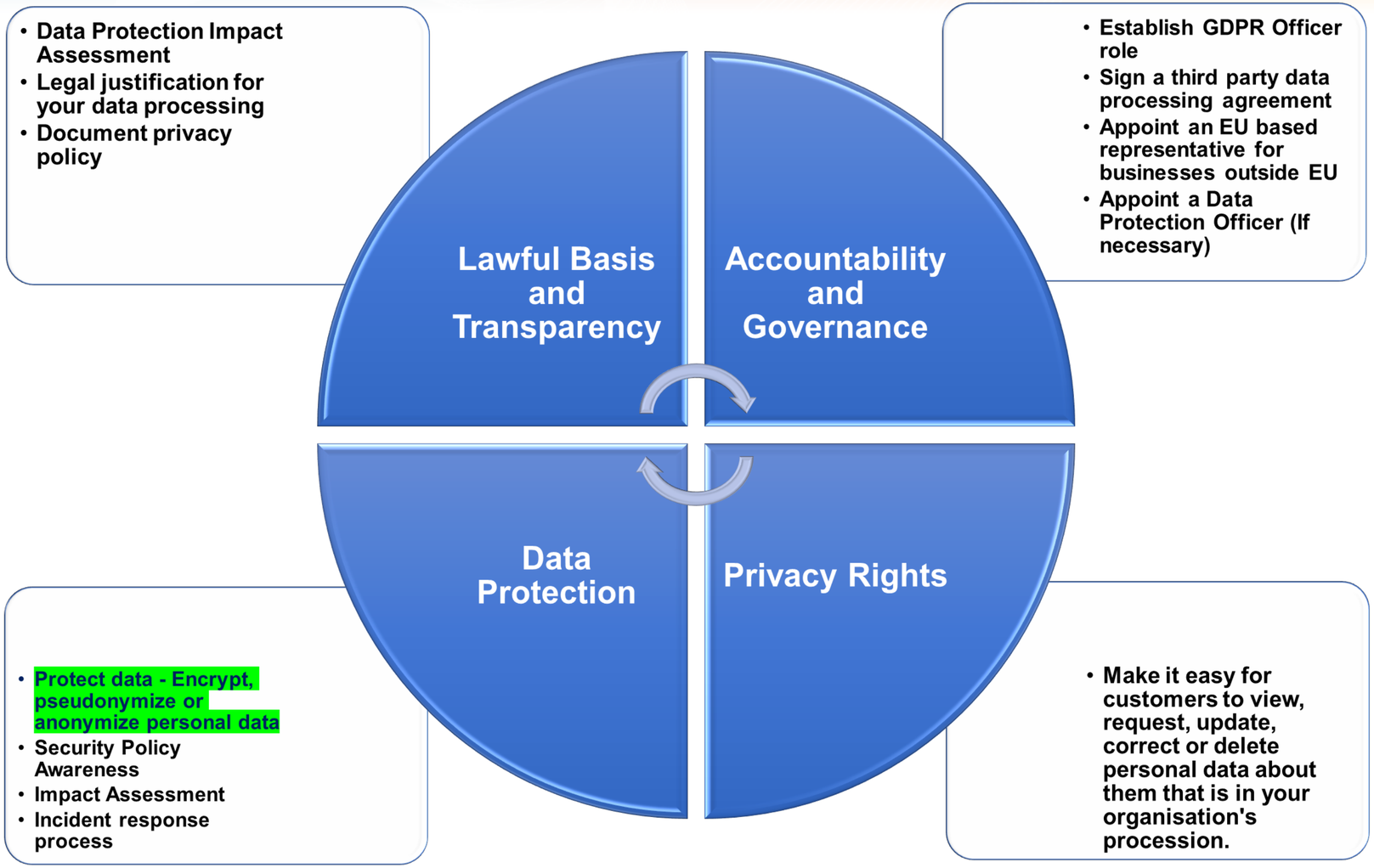
Output:

Below are GDPR data protection controls to be addressed as part of OSS level :

- Take data protection into account from product development to data processing.
- Encrypt, pseudonymize or anonymize personal data *whenever possible*

For more detail on the OSS GDPR scope - https://github.com/mojaloop/project/issues/1225

Mojaloop GDPR Scope



For more detail on the scope - https://github.com/mojaloop/project/issues/1225

mojaloop

1. Docker Security Enhancements.
2. Benchmarking of our adopted open source container vulnerability tools against commercial tools.

By Lewis Daly (Crosslake) and Victor Akidiva (Modusbox)

Container Security Enhancements

1) Added custom policy and tooling for evaluating Mojaloop container scans:
   a) Policy evaluation based on Docker CIS Benchmark best practices
   b) “Vulnerability Diff” A tool for comparing found vulnerabilities between Base Docker Images (**`node:12.16.1-alpine`**) and Derived Docker Images (**`Mojaloop/central-ledger:v9.5.0`**)
      See https://github.com/mojaloop/ci-config#container-scanning for more information

2) Started implementing Dockerfile best practices as guided by CIS Benchmark
   a) Focus on runtime pod configurations
   b) Key findings:
      i. Enable auditing on docker daemon and critical docker files. Do not mount sensitive system dir’s are not mounted in containers
      ii. Configure AppArmor profile
      iii. Restrict traffic between containers on default bridge
      iv. Ensure live restore is enabled. Add health check instructions to containers
      v. Restrict containers from acquiring new privileges. Do not run containers with root privilege.
      vi. Run containers with specific users other than root. Restrict ownership of specific docker files to root.
      See https://github.com/mojaloop/project/issues/1082 for more detail on the Dockerfile security recommendations

3) Focus on tightening existing security tooling, without slowing down developer workflow
   a) Better Slack build notifications
   b) Shared CI config that can be updated across all repos

Benchmark adopted Open Source Container Vulnerability Managements Tool vs Commercial Tools

Goal: To establish if there are any significant gaps between the open source tools we have adopt and the commercials versions specific to secure container configuration and vulnerability management

- Tests done with out of the box settings.
- Both commercial tool and community tool give vulnerabilities identified output
- Community
  - On average tool listed more vulnerabilities listed
  - Commercial tool offer neater reports with easy to prioritise options
  - Limited ability to connect to proprietary repositories.
  - Manual reporting i.e. extract and compile by yourself
- Commercial tool
  - Threat tracking as well as dashboards to track trends as well as reporting
  - Integration to various repositories (docker, quay, gcr)
  - Auto reporting including scheduled alerts
  - Advanced features available

With workarounds we feel community tools can comfortably support an open source CI/CD process.

Comparison Summary

AREA	OPEN	COMMERCIAL
Vulnerability Database & Updates	Open source vulnerability database e.g. https://docs.anchore.com/current/docs/overview/feeds/	Proprietary feeds
Out of the box Features	Basic features. 1. Manual Scanning 2. CI/CD integration 3. On prem and Cloud support 4. Additional features available on Enterprise tool (commercial)	Basic features. 1. Manual Scanning 2. CI/CD integration 3. On prem and Cloud support 4. Enterprise features such as workflows, automations, exploit testing, trend analysis, linkage to enterprise vulnerability tracking
Vulnerability Findings effectiveness	Vulnerability details with CVE	1. Vulnerability details with CVE 2. Exploitability and details of available exploit (some tools)
Liability	Limited liability due to open source	Enterprise support cover
Hidden Costs	Many open source projects provide active forums, up-to-date documentation and detailed tutorials, but there is a trend towards charging for dedicated support, something to watch out for if your budget is tight.	Cost as quoted at time of purchase. Usually license + professional services + AMC and support (other costs may include hardware etc.). Additional features may be licensed separately.

In Conclusion, there are no notable gaps between our adopted open source toolset against leading commercial tools that can leaves exposed. We will continue using the current tools and updating them as the threat landscape changes.

mojaloop

1. Security Architecture Review\Update.
2. HSM Deployment Recommendations.

By Pedro & Godfrey

Security Architecture Review & Standardization (Top Down Approach)

1. **Switch Security**
    a) Intra switch security
        i. How services authenticate and authorise when talking to each other
    b) Inter switch security (multiple switches scenario)
        i. How multiple switches authenticate and authorise when talking to each other
    c) Operator authorisation and authentication (identity management)
2. **Switch Infrastructure Security**
    a) Message topics and database authorisation
    b) Platform secrets (cluster) & Gateway secrets
    c) Data Protection - Kalka and Database Security
3. **Switch <-> FSP Security**
    a) How to secure FSPs authenticate and authorise when talking to each other
    b) Callbacks from the switch (Check for pain points from existing implementations)
4. **Secure Mechanisms for key/secret generation, storage, verification and invalidation**
    a) Provide a standardized way with HSM been the underlying support technology implementation.

Our approach is to provide OSS based solutions as pluggables with standard provided implementations so that implementers do not have to touch the OSS code in provisioning all these services.

Our HSM Deployment Approach

1) **Adhere to best practice HSM design, deployment & operational practices:**

   a. Key Management System support (Master Keys, Session Keys, CA, Certificates, etc..), Secure management processes (SOD's) & systems authentication
   a. FIPS 140-2 Level 3 Compliance, PCI DSS, CC and other regional standards
   b. Secure Crypto Processing (Payment vs General Purpose Functions),
   c. Infrastructure & Operational - Performance, High Availability, Scalability, etc..

2) **Support multiple open cryptographic standards:**

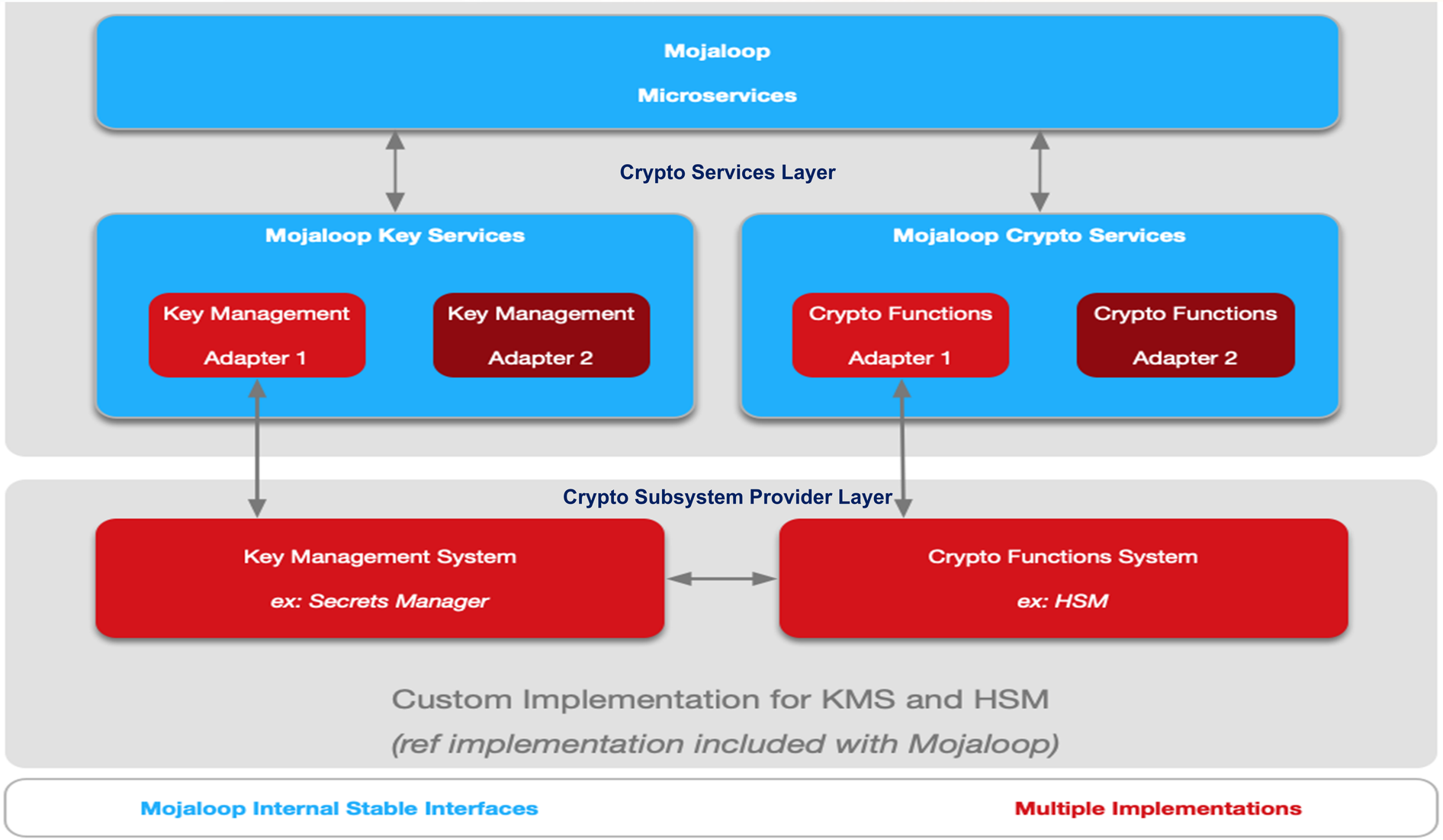
   a. Support both KMIP and PKCS#11 as the standardized interface to the KMS and HSM resources (frontend and backend, respectively) and provide a strong basis for HSM vendor command independence and decouple from vendor SDK's.

3) **Adopt open architecture integration methods:**

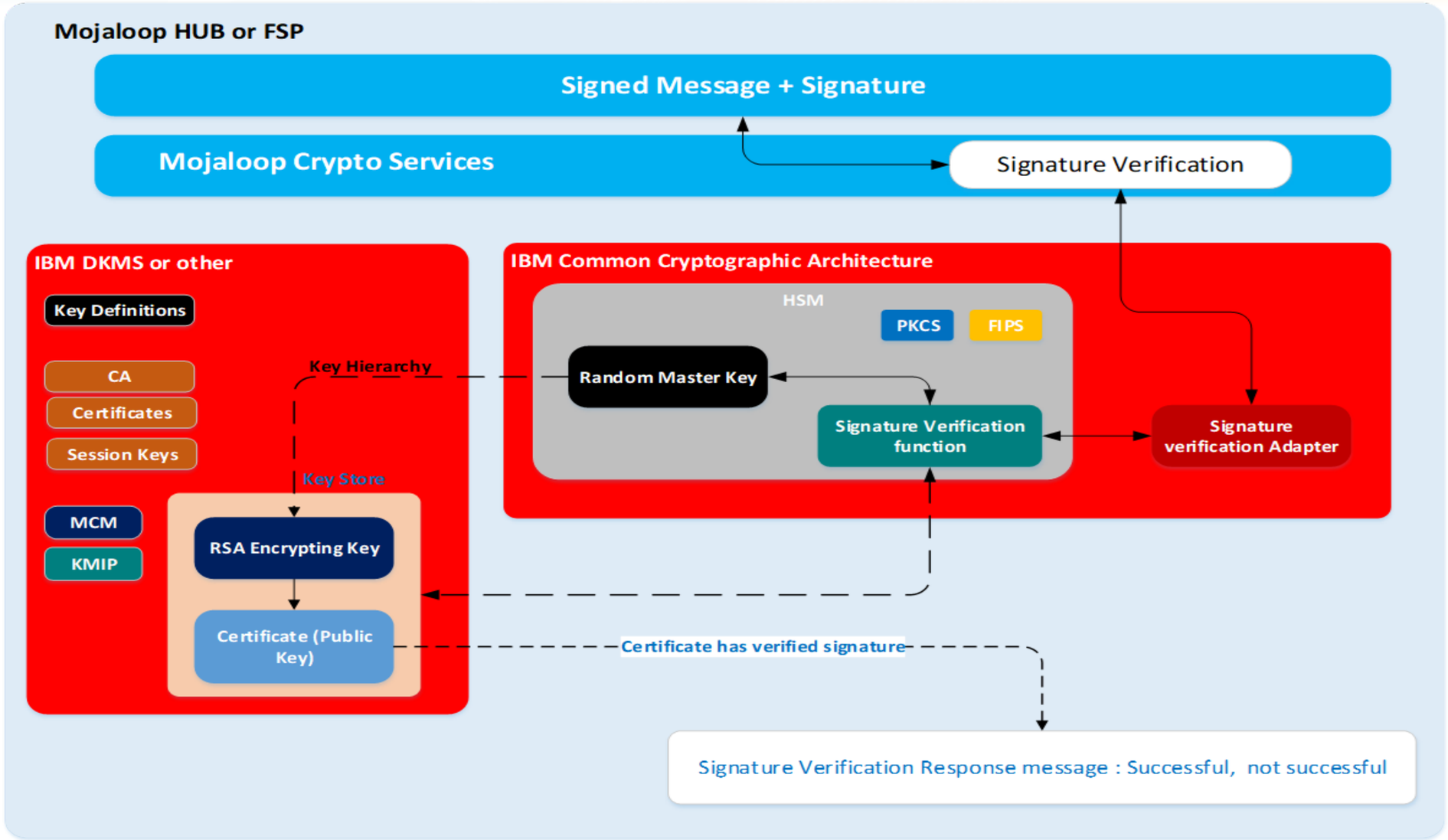
   a. Implement a bridge\adapter architecture for internal services consume security functionality from an internal service.
   b. Provide an open source solution, which in turn can implement several adapters for each external interface (for example talking to a specific HSM implementation or standard).

From core architecture perspective, HSM deployment should be flexible and open enough to support multiple use cases with minimal or no changes to the OSS code base

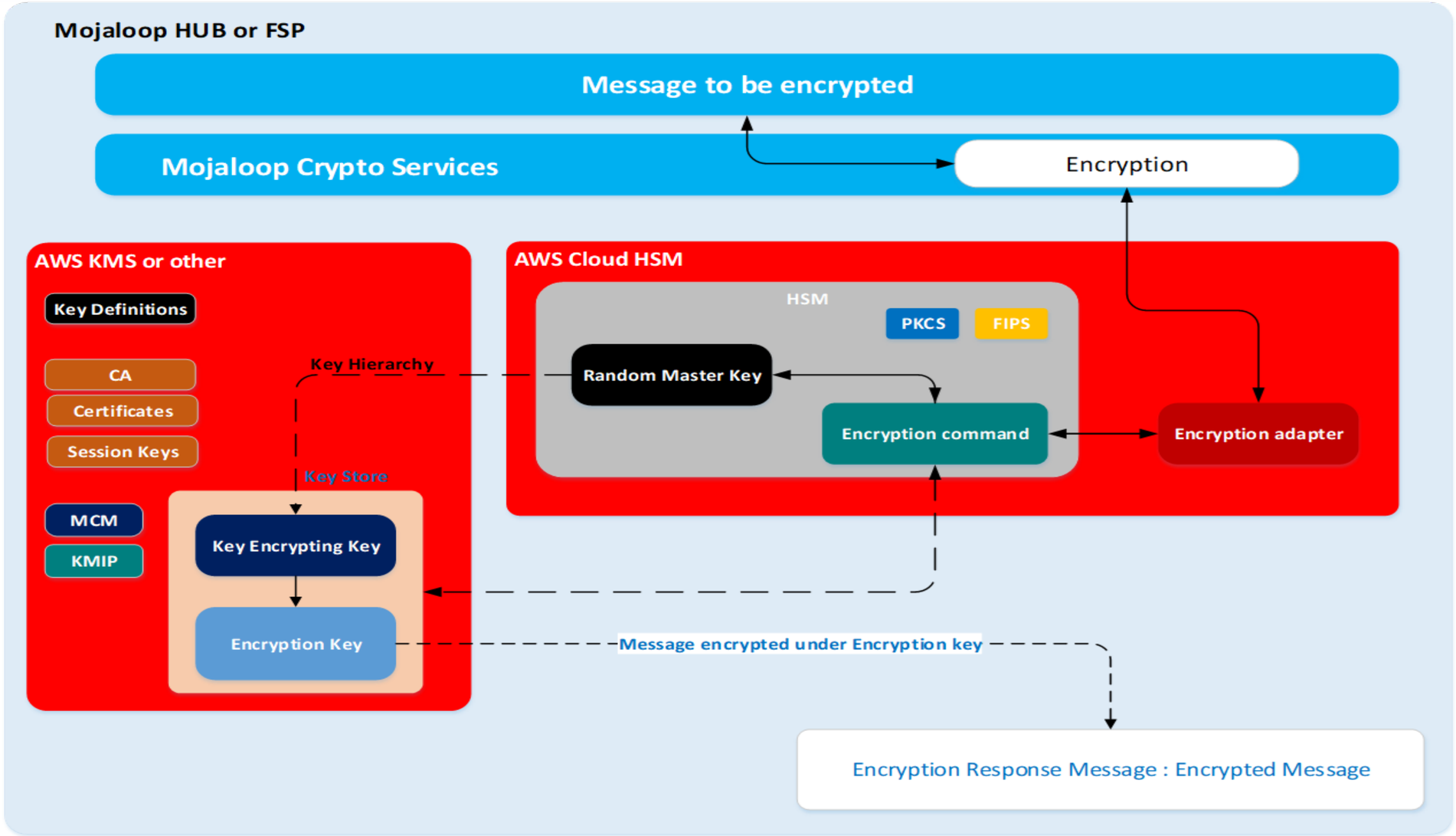
Crypto Integration Model



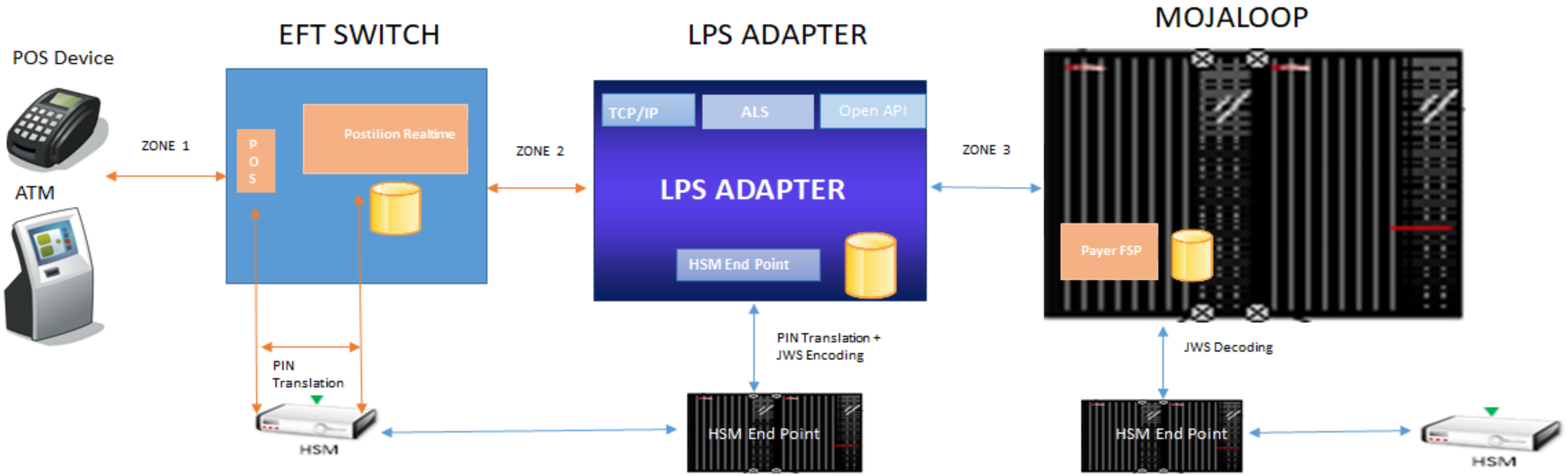
Use Case 1 : Signature Validation Service



Use Case 2: Encryption Service



Use Case 3: HSM Integration for POS Mojaloop OTP Encryption



The use case entails the encryption of the OTP at the source (POS or ATM), transmit it securely in the standard ISO Pin Block field (DE 52), ensure the security of the OTP by performing crypto operations through the HSM end point and then securely transmit it using JWE signature through to Mojaloop.

mojaloop

muito obrigado

Questions and Comments

Please join us in the code improvement workstream breakout session tomorrow for inputs and deliberations in setting PI 10 Objectives

“ Work is not done by a magnificent plan or strategy; work is done, when is done, and done by people.”

mojaloop

Proposed PI 10 Objectives

1) HSM Implementation - Design, POC, Crypto API\Adapter & Use Case Implementation (ATM\POS OTM and Signatures).
1) Plan Implementation of the GDPR scope – Perform an assessment and Investigate options for addressing data protection requirements.
1) Baseline Mojaloop Security against PCI DSS Standard - Gap Analysis and Implementation Plan
2) Threat Modelling – Focusing on Parties, Quote & Transfer processes.
3) Develop new security standards for:
    a. Intra Switch Communication - How services authenticate and authorize when talking to each other
    b. Data Protection - Kalka and Database Security
    c. Key/secret generation, storage, verification and validation

NB. No input received so far from implementations projects however the door is always open.

mojaloop

Backup Slides

GDPR Requirements for appointing a DPO

Under the GDPR, appointing a Data Protection Officer (DPO), is mandatory under three circumstances:

1) The organisation is a public authority.

2) The organisation is a public body (except for courts acting in their judicial capacity).

3) The organisation's ***core*** activities consist of data processing operations that require ***regular*** and ***systematic monitoring*** of data subjects on a large scale.