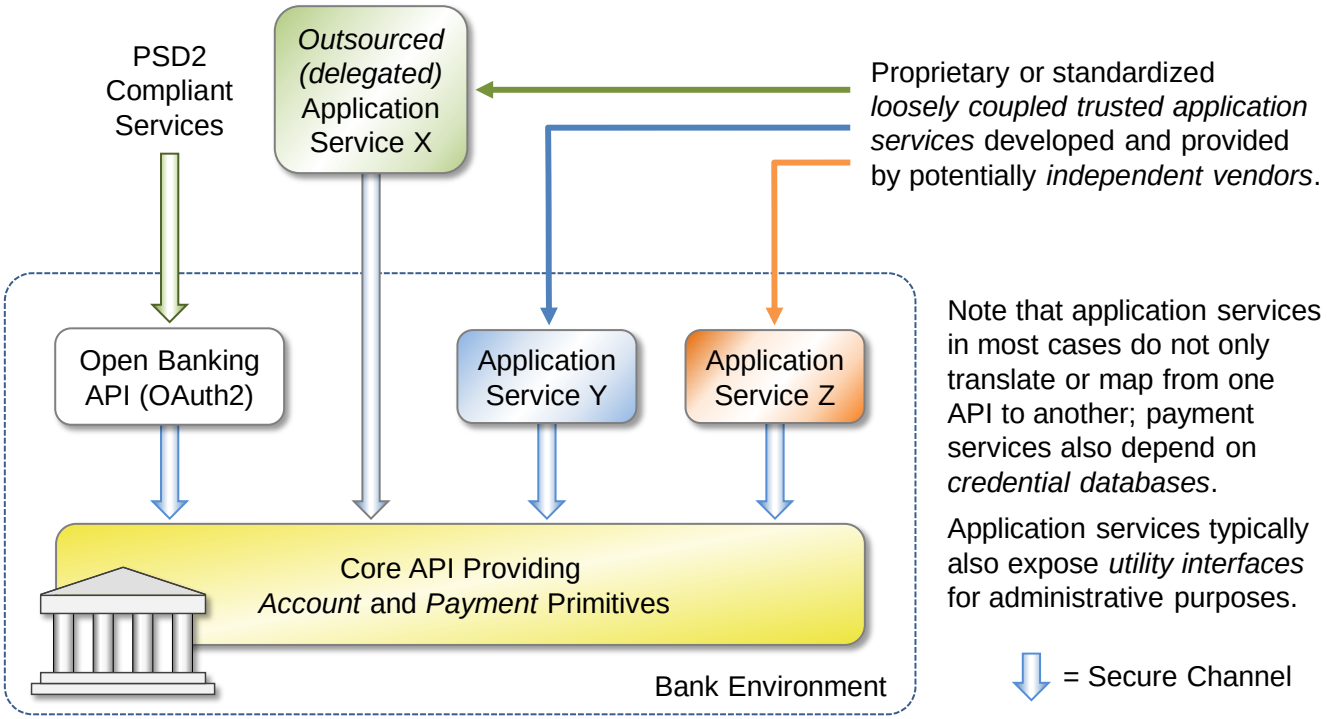
Revised Open Banking Architcture

Background: As you may have noted the combination of the evermore popular digital wallets and Open Banking do not seem to happen. The reason for that is quite simple: the integrated SCA concept in current Open Banking APIs interferes with wallets like Apple Pay that have their own SCA solution.

To make things worse, existing Open Banking systems are based on a *single-level, monolithic architecture* making the addition of new features very costly and cumbersome, as well as effectively blocking third party development.

This document outlines another solution which could be illustrated like the following:



The described architecture builds on a *layered model* for Open Banking APIs, where a fixed Core API, supports multiple, *trusted application services*. Since the Core API is a *privileged* service, the Core API is not exposed directly to external services (the trusted application services function as *mediators*). To maintain system integrity, the trusted application services and the Core API, communicate through *mutually authenticated* TLS channels. Judicious use of *logging* is highly recommended. Note that the Core API does not authenticate *users* (this is supposed to be performed by the trusted application services).

The Core API consists of JSON based REST services, enabling *loosely coupled* application services.

Summary:

- The non-monolithic nature of this arrangement simplifies Open Banking API documentation, implementation, and testing.
- Loosely coupled application services can be built on *vendor specific platforms*. The loose coupling also enables application services to be shipped as boxed systems, docker images, or virtual machines.
- The described architecture permits support for Core API *sandboxes*, making application service development in reach for any entity. Using sandboxes, application service iterations become simple and fast.

Write Once – Deploy in Any Compliant Bank

A.Rundgren 2024-01-01