Endorsement/RV Distribution to Verifiers

Defining an API for RATS and using Veraison as a prototyping platform

Paul Howard, Arm

CONFIDENTIAL COMPUTING CONSORTIUM

VERAISON

Who Am I?

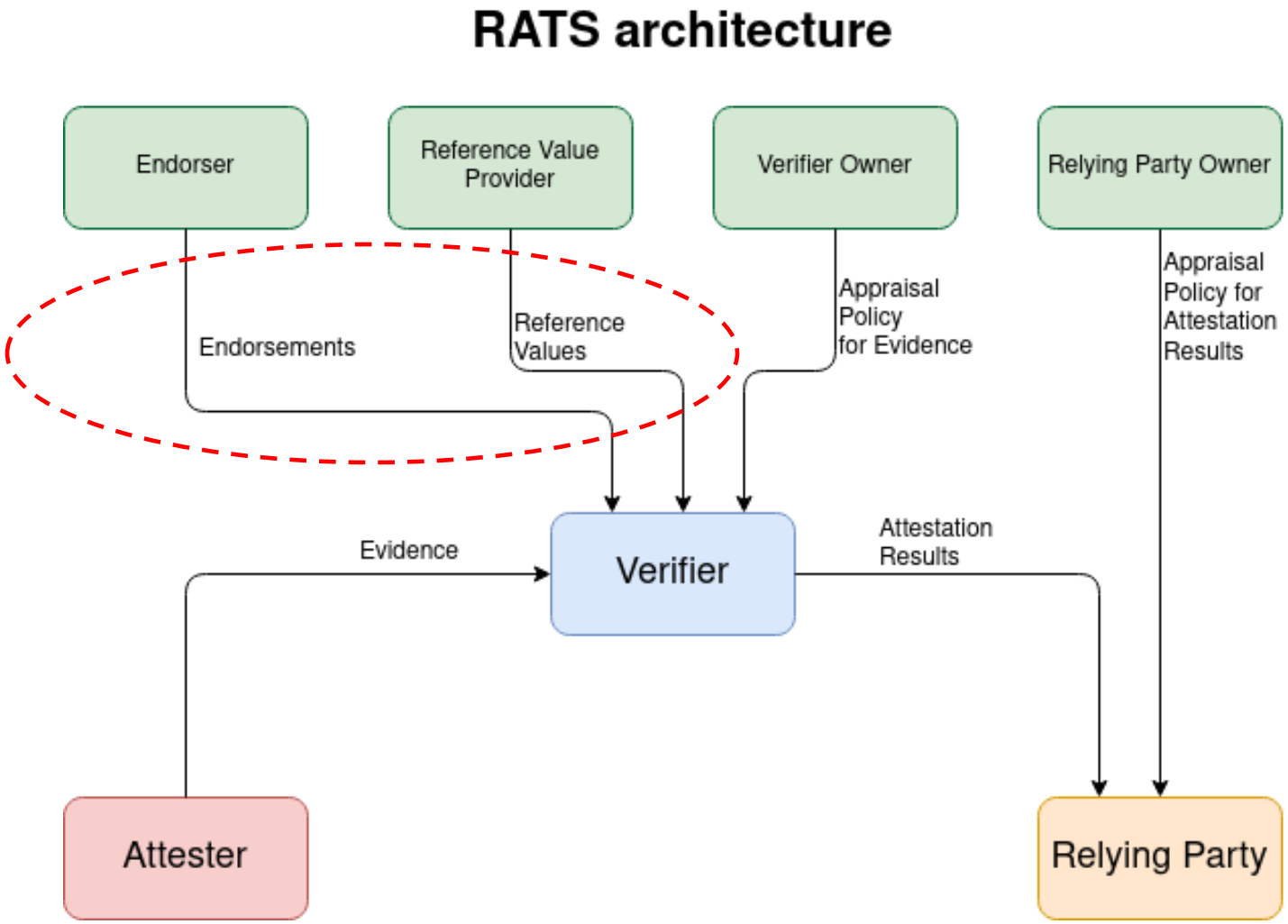
Paul Howard
Principal System Solutions Architect at **Arm**
paul.howard@arm.com
https://confidentialcomputing.slack.com
https://www.linkedin.com/in/paulhoward4/

@paulhowardarm

Some Context

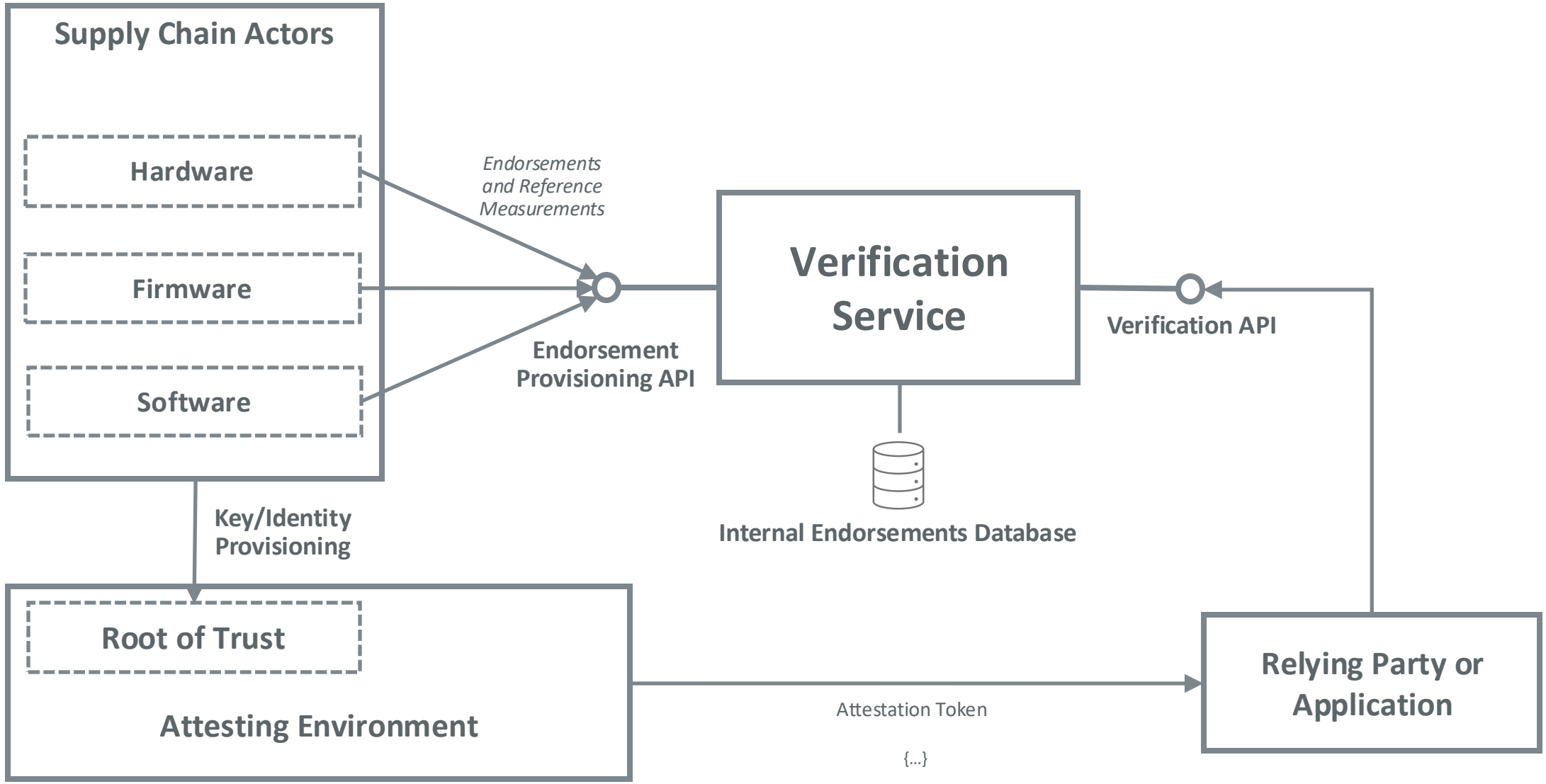
- Attestation is complex with many roles, components and stakeholders – from hardware supply chain all the way to running end-user applications
- Fragmentation is a risk and a barrier to success
- Many standardization efforts are already underway in collaborative communities
- But there's room for more!
- We can analyze the RATS model to determine opportunities to propose standards
- And use open-source software to create PoC implementations and building blocks – working software, not just documents

Problem Space: Endorsement/RV Distribution in RATS



https://datatracker.ietf.org/group/rats/about/

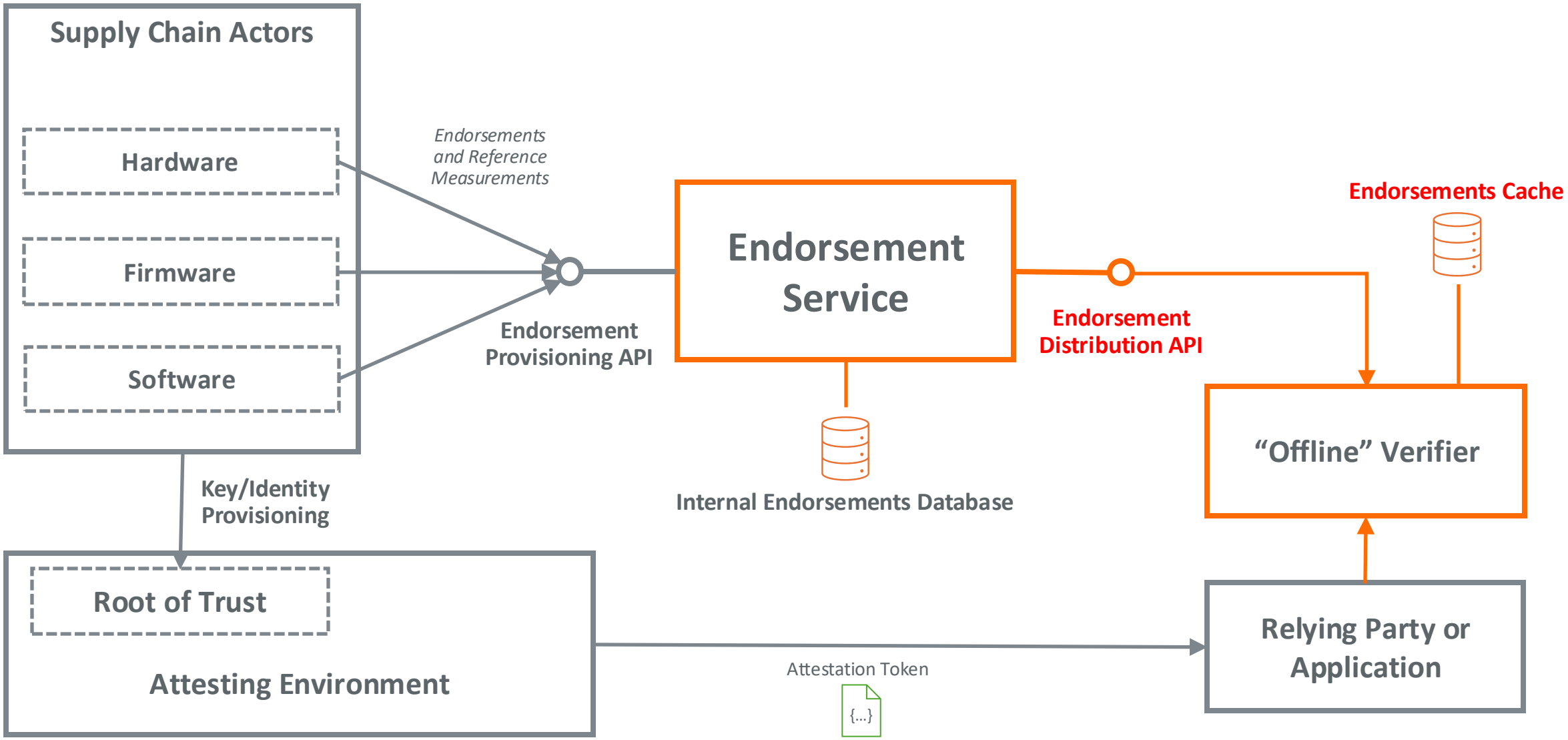
Verifier Service Pattern (Exemplified by Veraison)



Assumptions of the Verification Service Model

- That there is an **intimate connection between the verifier and the supply chain actors**, such that the provisioning and verification can be coalesced into the same service
- That the verifier service is **always available** when the Relying Party needs to verify an attestation
- That the Relying Party can **afford the cost** of an online service call for every verification
- These assumptions can make the verification service model **unsuitable for some deployments**

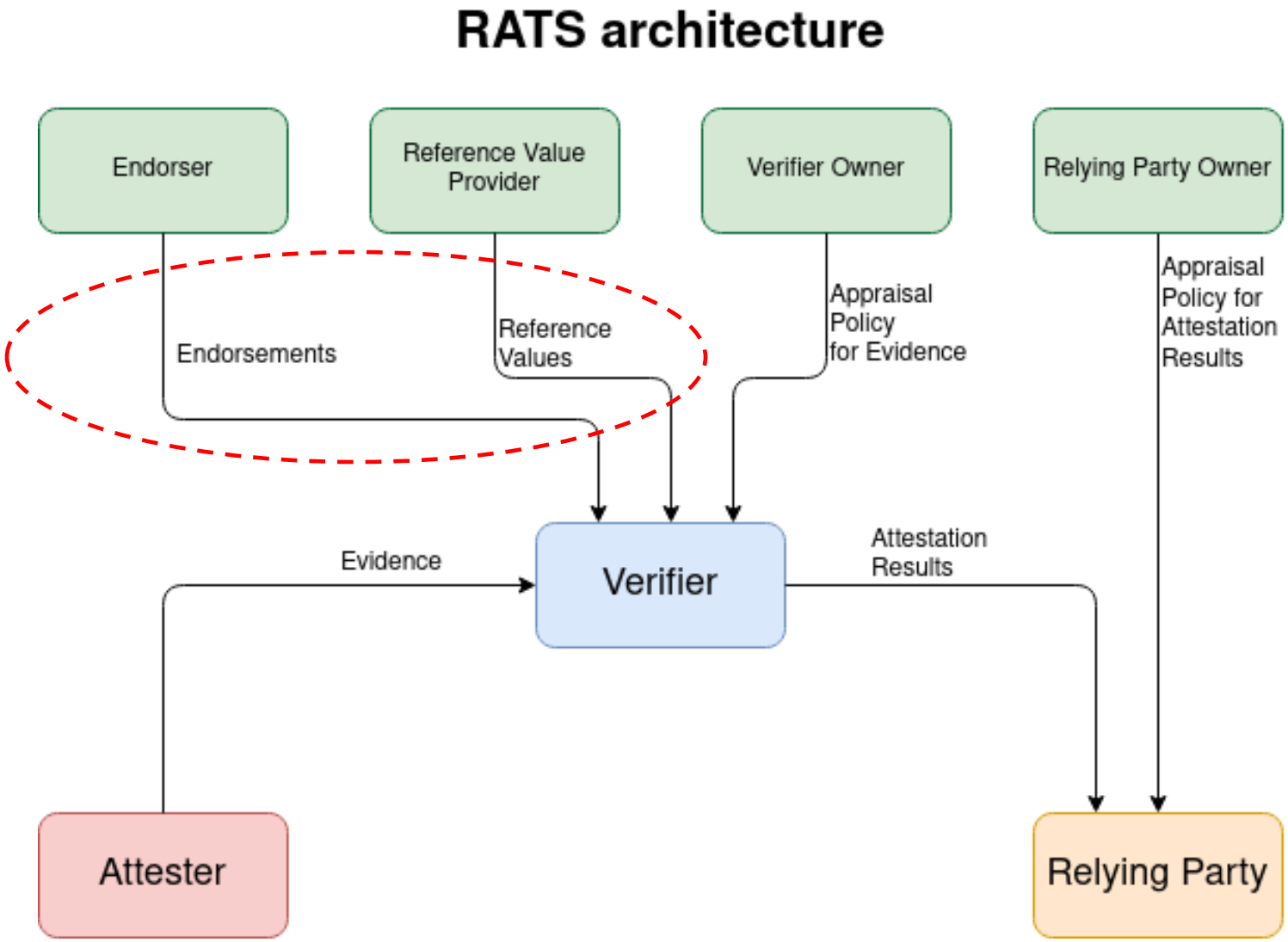
Endorsement Service Pattern



Characteristics of the Endorsement Service Model

- **Disaggregates** the provisioning and verification processes
- Allows **levels of indirection** between the supply chain actors and the verifier
- The verifier **can make choices** about when to make online API calls based on a flexible caching policy
- Verifier can be available via **local procedure call or even just a library function call**
- Example of **library-based verifier**: https://github.com/veraison/rust-ccatoken

Endorsement Distribution in RATS Architecture

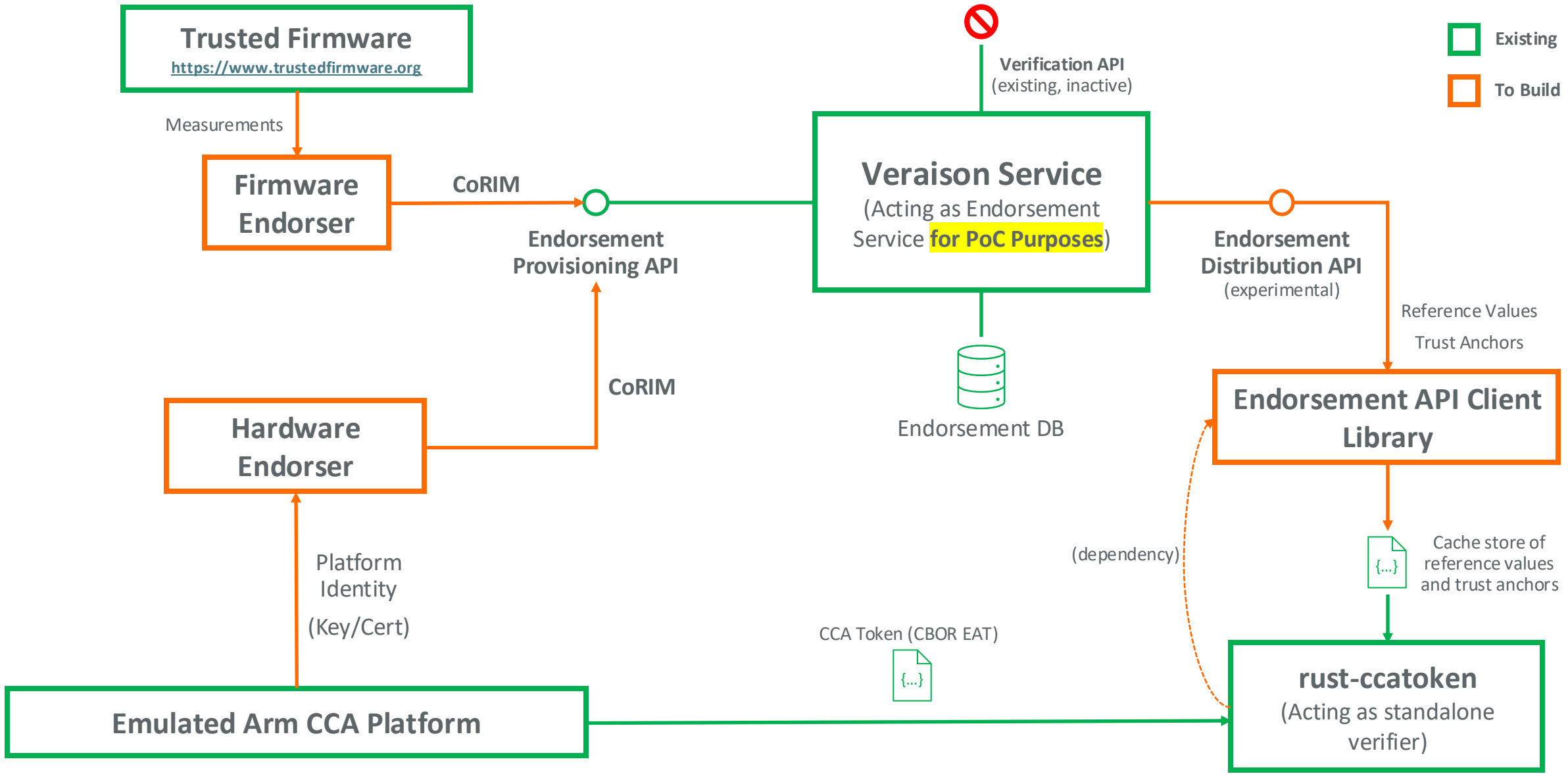


- **Fragmentation** is a risk, especially on Arm platforms, where Arm IP is consumed by multiple distinct silicon vendors
- Conveyance of endorsements is **within IETF RATS WG sphere of concern**, and therefore an attractive pathway to standardization
- Arm is seeking collaborators for an **IETF Internet Draft document** for an endorsement distribution API

Why Veraison is the Ideal Experimentation Platform

- Already has the capability to ingest endorsements/RVs from supply chain actors and to store them internally, which positions it well for prototyping an API to distribute them to consumers
- The **rust-ccatoken** repo offers a good model for a library-style verifier, which positions it well to be the consumer of the new API
- Veraison was originally intended for the construction of verifier services, but it could be extended to act as an endorsement service, initially for prototyping, but potentially as an upstream feature

Proposed Prototyping Architecture



Endorsement Distribution API: Some Design Considerations

- **Trust Model**
  - Centralized – the endorsement service is trusted by clients, with implicit trust in supply actors
  - Distributed – the endorsement service is trusted by clients,
- **Interaction Model**
  - Frequency and granularity of API calls
  - Observer pattern (registering for change, pub/sub)
- **Data Model**
  - Things that are common:
    - The notion of a measured component
    - Timestamp of most recently-cached data
  - Things that are specific (example):
    - Arm CCA platform reference values (rust-ccatoken has a CDDL model for this)

Help Is Needed!

- Become stakeholders in the API design:
  - Put forward names, organizations and email addresses for inclusion as design co-authors on the IETF Internet Draft document
  - Be willing to contribute or review content as the draft evolves
  - Take part in community (public) discussions on suitable meetings and chat forums
- Hands-on involvement in prototyping (optional)
  - All code must be contributed publicly to open-source projects according to their existing license terms

Thank You!

Discussion