TACRA:
Trustworthy Acquisition of Credentials using Remote Attestation

IETF 127: RATS and LAMPS Working Groups

Mark F. Novak, JPMorgan Chase & Co.
Henk Birkholz, Fraunhofer Inst.
Serhii Nikolaichuk
Michael Richardson

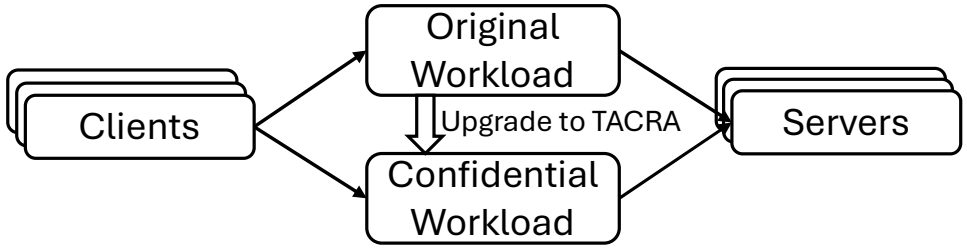
https://datatracker.ietf.org/doc/draft-novak-rats-tacra/

https://datatracker.ietf.org/doc/draft-novak-lamps-tacra-est/

Problem Framing

- All data in computing is exchanged in one of three ways:
  - Between humans and machines
  - Written to logs
  - **Between workloads** ← TACRA fits here
- Workloads must authenticate to each other
- For that they need credentials
- TEEs protect these credentials in use, alongside any other data entrusted to the workload
- Provisioned credentials cannot be visible to the Control Plane, else in-use protection of them becomes worthless
- Remote Attestation can be used for secure credential provisioning, using Control Plane as a conduit only

Key point: credentials presented by the workload to its clients and servers are indistinguishable from those that it was presenting before the upgrade



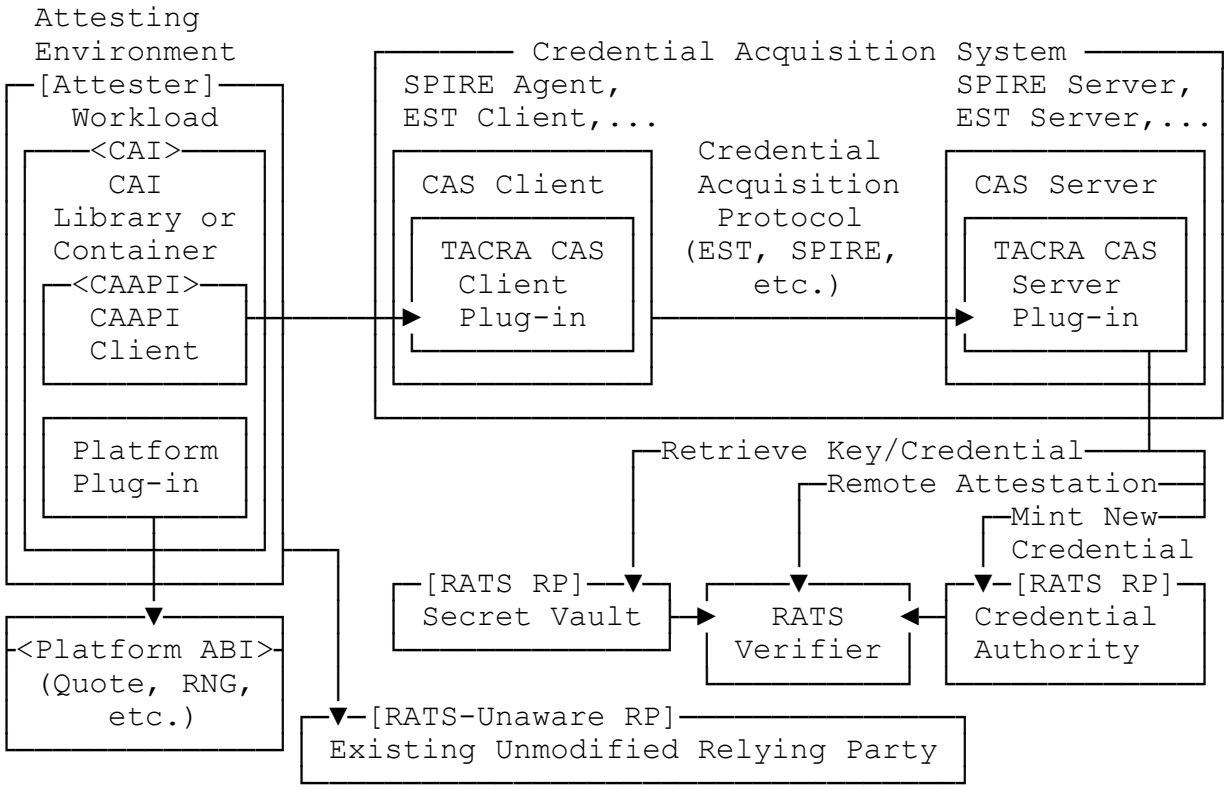
- Most existing credential acquisition systems (CAS) do not support Remote Attestation
  - EST, SPIRE, ACMEv2, …
- Overwhelmingly, existing clients and servers are RATS-unaware: they do not, and will not understand Attestation Results
- Success of any technology is ultimately measured by its adoption
- Inability to upgrade existing workloads to get their credentials via Remote Attestation is a significant adoption blocker for enterprises
- Solution: TACRA Architecture

TACRA Requirements

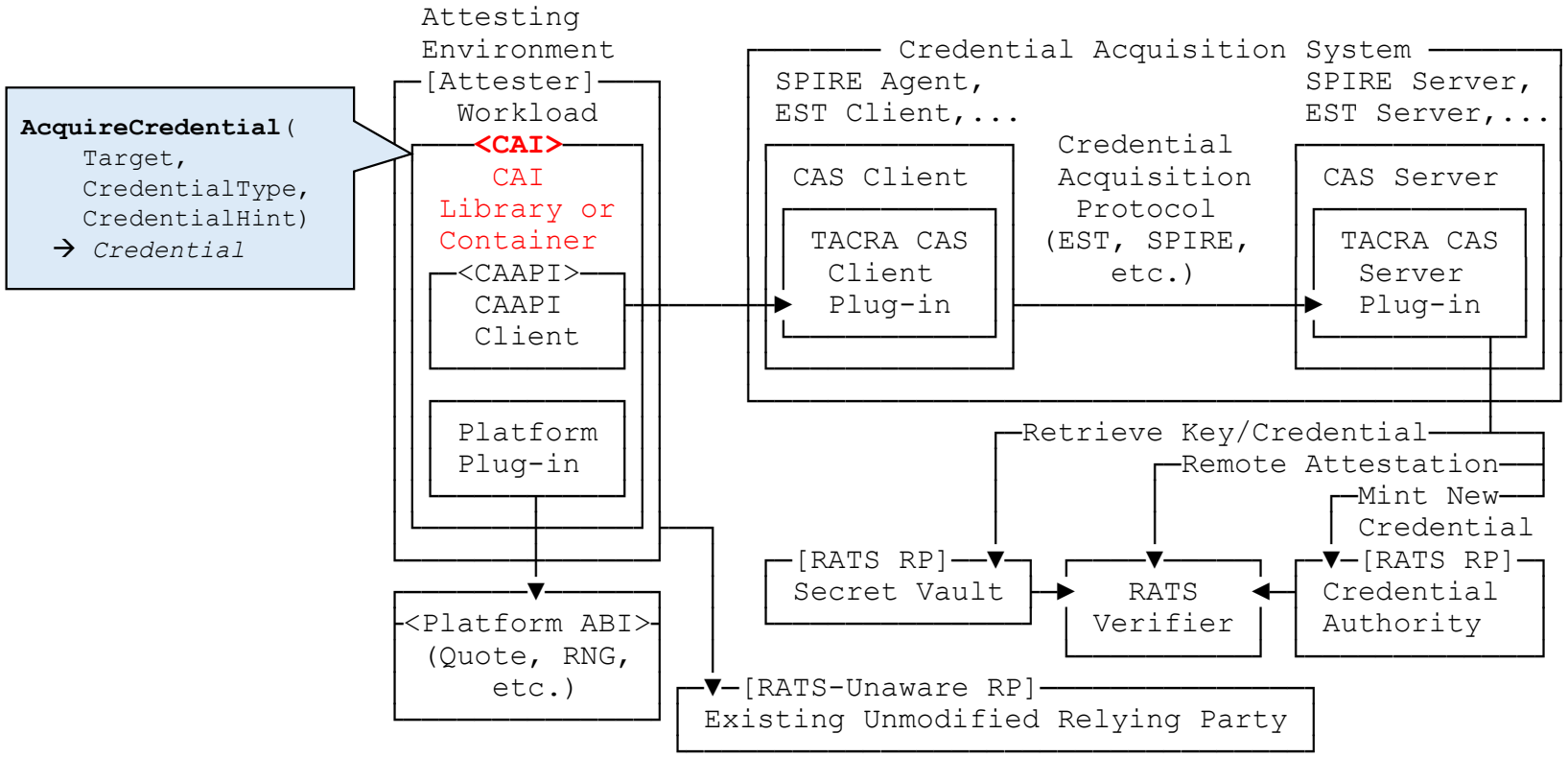
1. Must support mechanisms for enrolling (minting new) as well as retrieving (pre-existing) credentials; the workload knows what Credential Type it will need when it launches, but discovers whether it will have to retrieve an existing or enroll a new credential at runtime.
2. Must support most current and future credential formats:
   - Proof-of-possession creds (X.509 Certificates, WIMSE WICs, WITs)
   - Bearer tokens (JWTs, API Keys, pre-shared keys)
   - Future formats through architectural extensibility
3. Should limit trust boundary expansion of the Attesting Environment to the minimum
4. Must support, transparently to the Attester, most current and future Credential Acquisition Mechanisms:
   - EST (RFC 7030)
   - SPIFFE/SPIRE
   - ACMEv2 (RFC 8555)
   - Future mechanisms through architectural extensibility
5. Must support Workloads utilizing different Credential Acquisition Mechanisms per-target
6. Must support both Background Check and Passport RATS modes, indistinguishably from the PoV of the Attester
7. Must support most current and future RATS Verifiers, Credential Authorities, and Secret Vaults, transparently to the Attester
8. Must not assume that the Attester has network access
9. Must be compatible with all existing and future Confidential Computing platforms meeting minimum requirements around secure cryptography and evidence generation
10. Should restrict visibility of fetched secrets to the Attester, specifically excluding visibility by the CAS Client and, if possible, the CAS Server

NB: It is not a goal, and, at any rate, it is not possible, to leave credential acquisition protocols (EST, SPIFFE/SPIRE, etc.) unmodified.

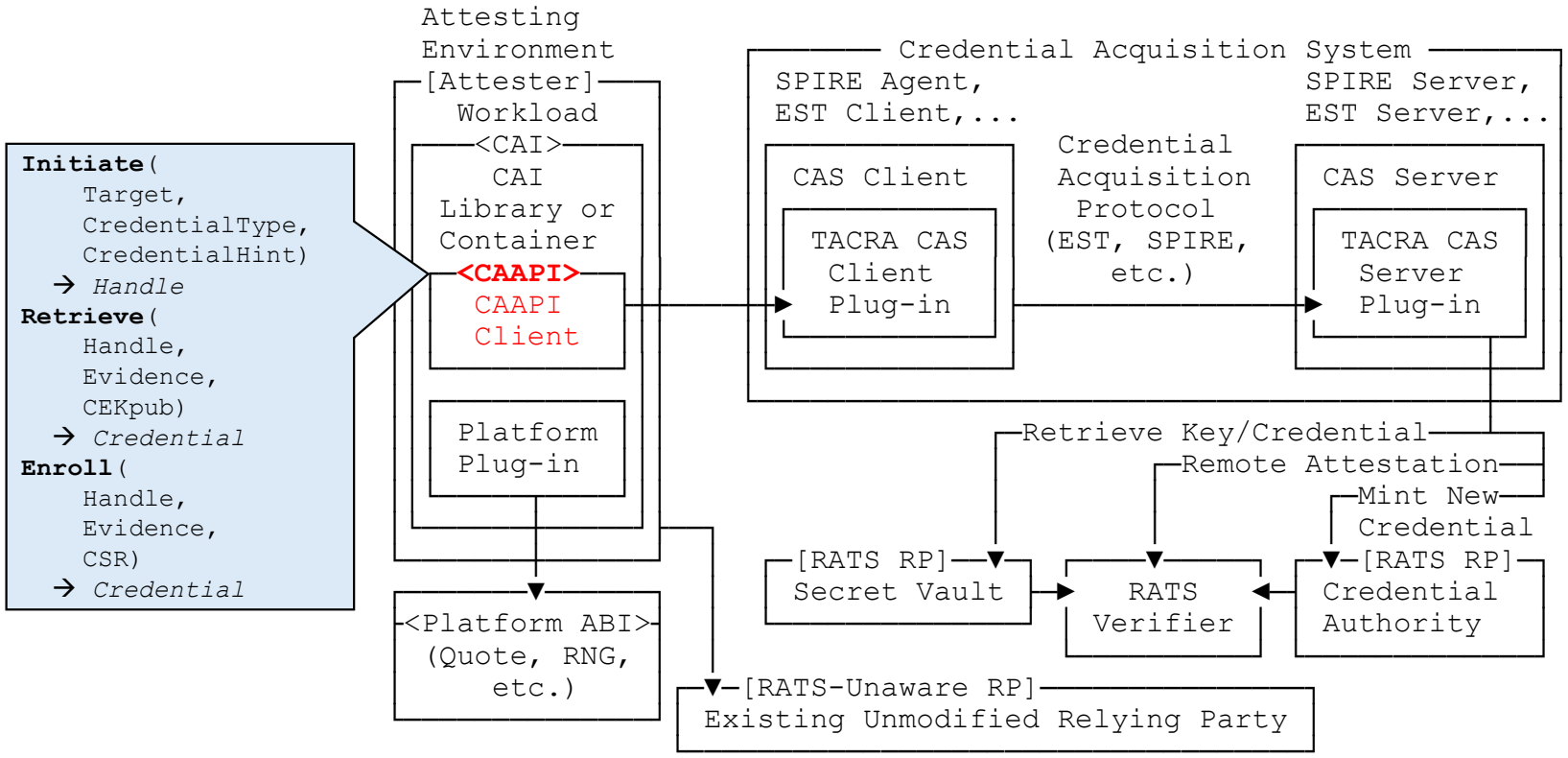
TACRA Architecture



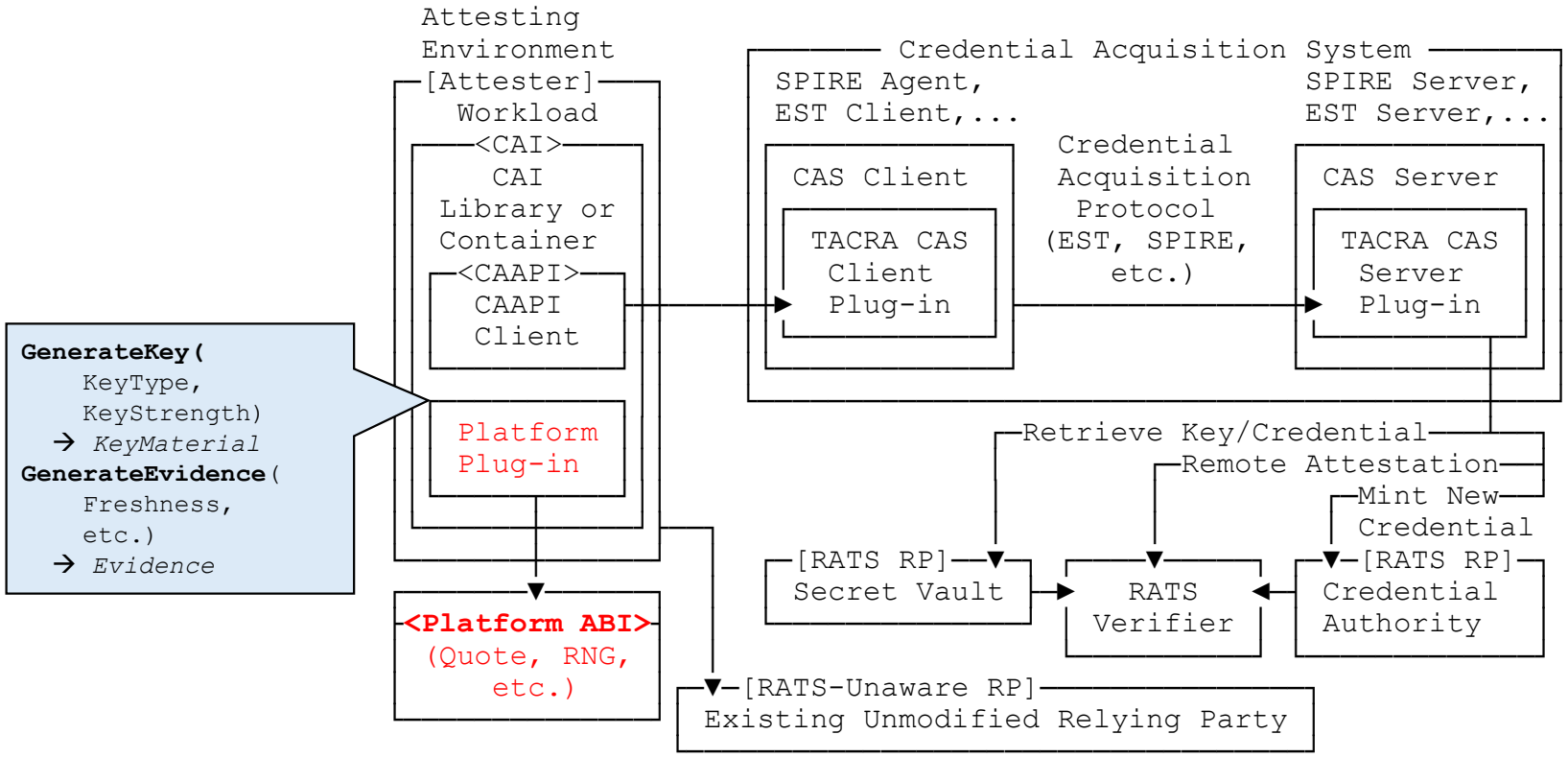
Credential Acquisition Interface (CAI)



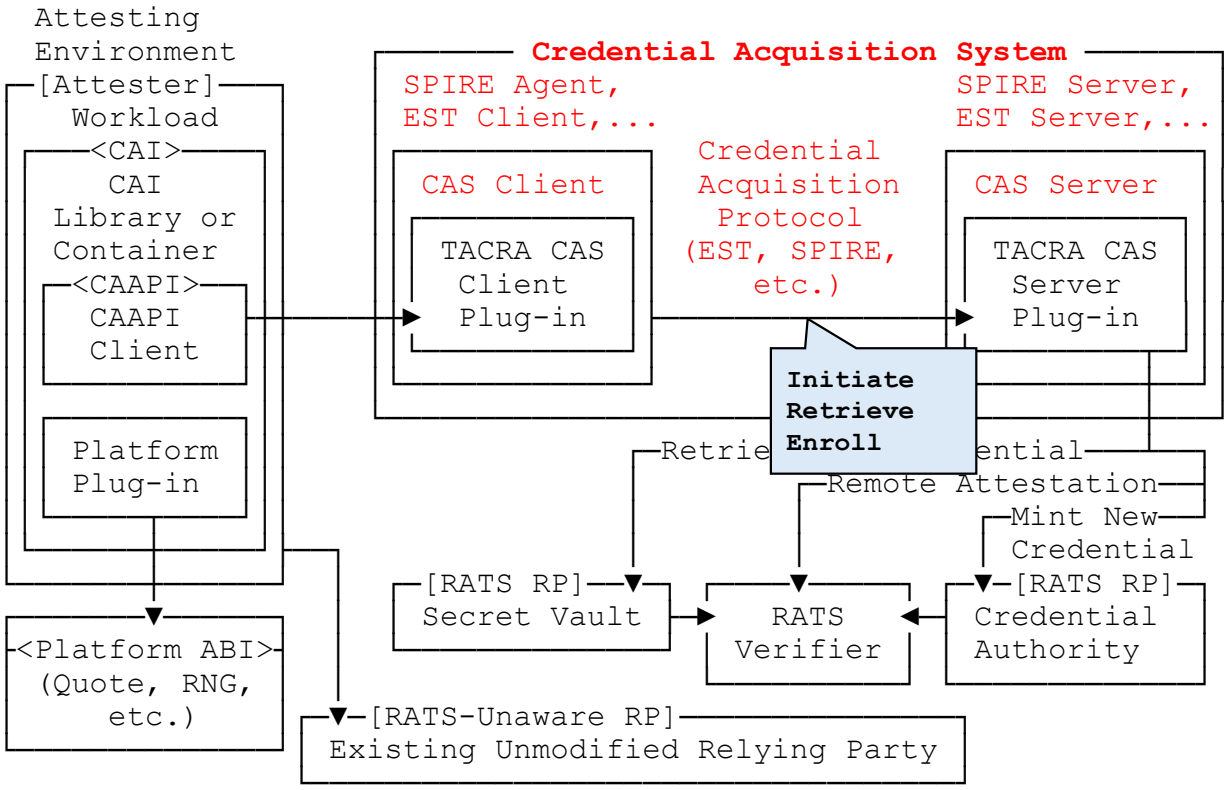
Credential Acquisition API (CAAPI)



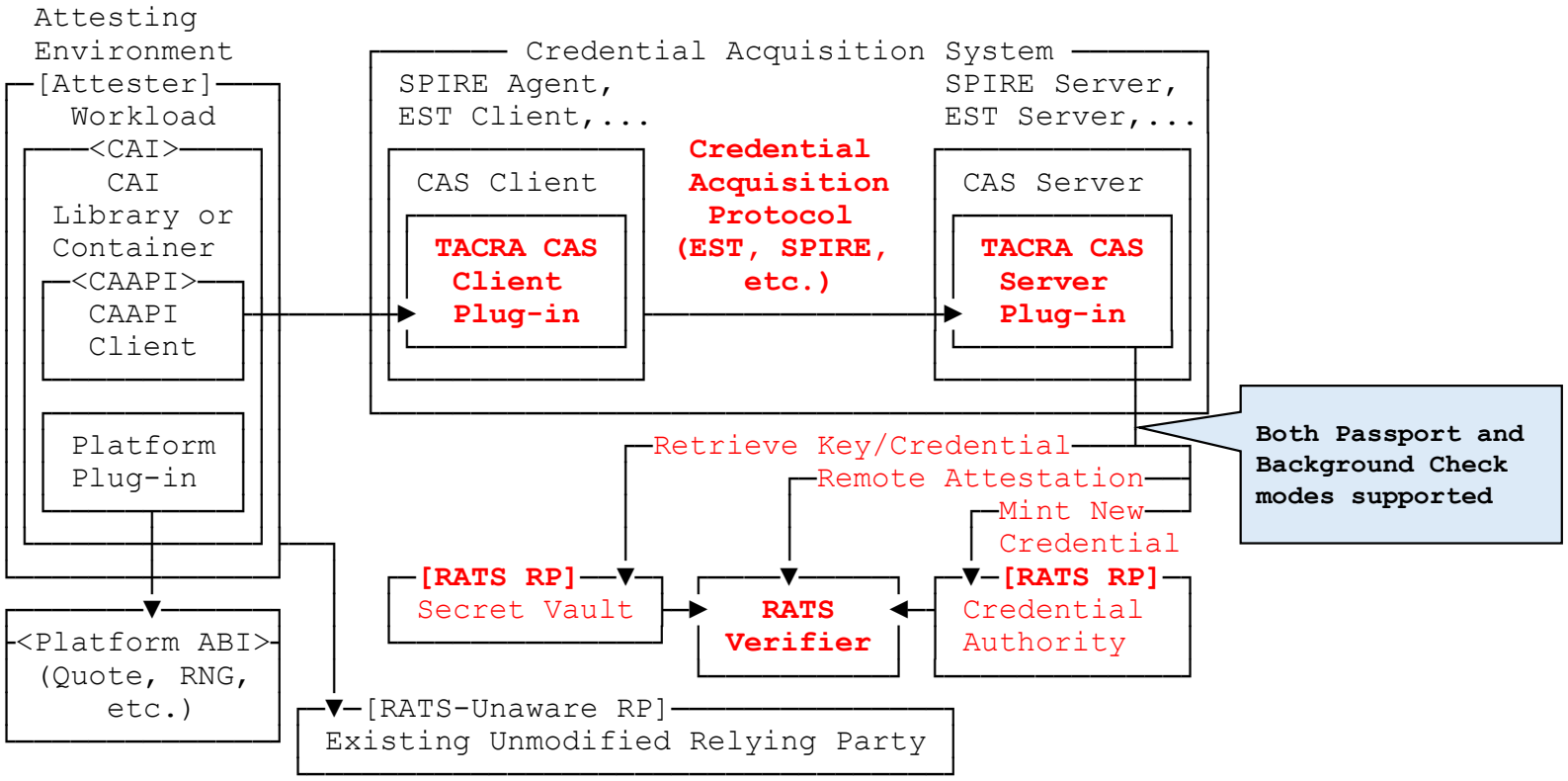
Platform Plug-in and Platform ABI



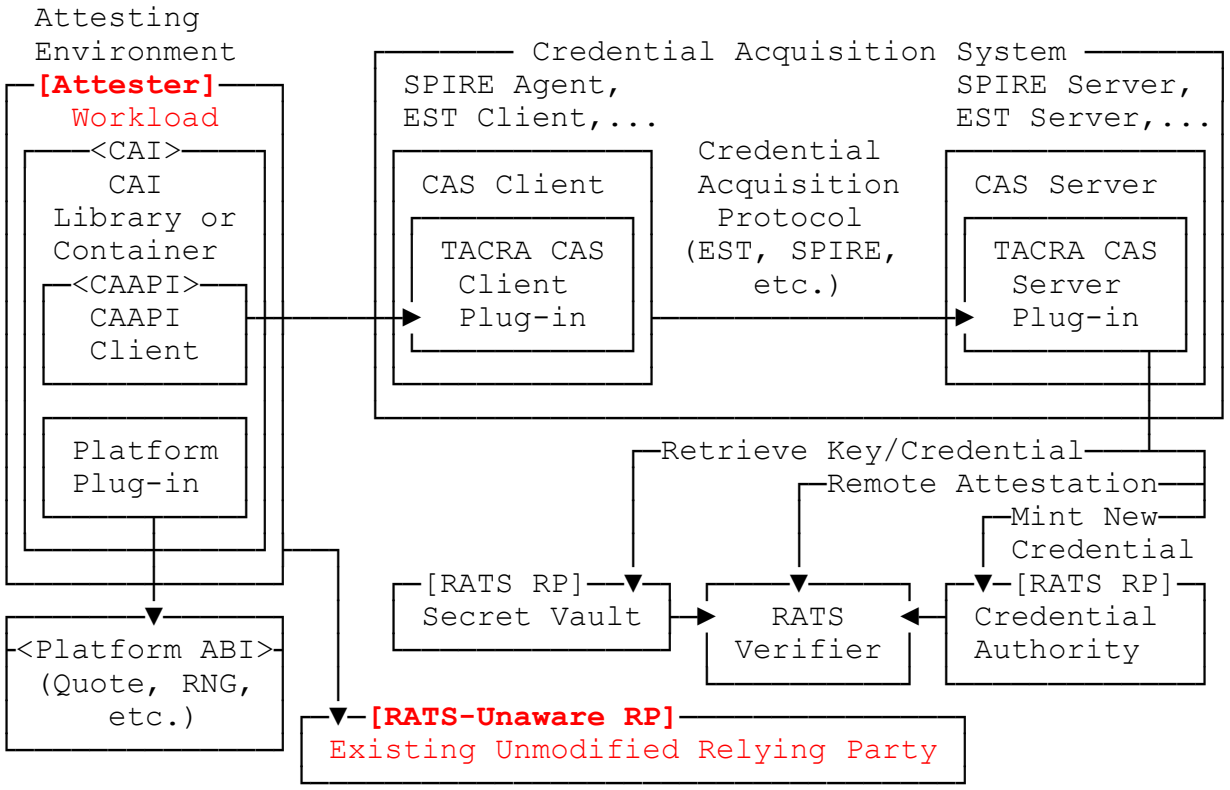
Credential Acquisition System (CAS)



CAS Plug-ins and RATS



RATS-Unaware Relying Party



Credential Acquisition System Modifications

EST

- New EST resources
  - attest-initiate
  - attest-enroll
  - attest-retrieve
- EST client sends new commands, does not get to see any secrets in the clear
- EST server handles new commands

SPIRE

- New Workload API
  - InitiateCredentialAcquisition
  - EnrollCredential
  - RetrieveCredential
- SPIRE Agent sends new commands, does not get to see any secrets in the clear
- SPIRE Server handles new commands

References

With special thanks to the Trustworthy Workload Identity SIG at the Confidential Computing Consortium.

- TWI SIG Charter: https://github.com/confidential-computing/governance/blob/main/SIGs/TWI/TWI_Charter.md
- TWI Requirements: https://github.com/confidential-computing/twi/blob/main/TWI_Requirements.md
- TWI Definitions: https://github.com/confidential-computing/twi/blob/main/TWI_Definitions.md
- This I-D: https://github.com/TheBankster/rats-tacra/
- EST with TACRA I-D: https://github.com/TheBankster/lamps-tacra-est/
- TACRA Implementation: https://github.com/TWI-Unified-Model/
  - Includes the reusable TACRA/CAI/CAAPI engine, EST and SPIRE implementations
  - Near-production quality code, not just a PoC