

Secure Wasm App Provisioning with VERAISON: Design Insights

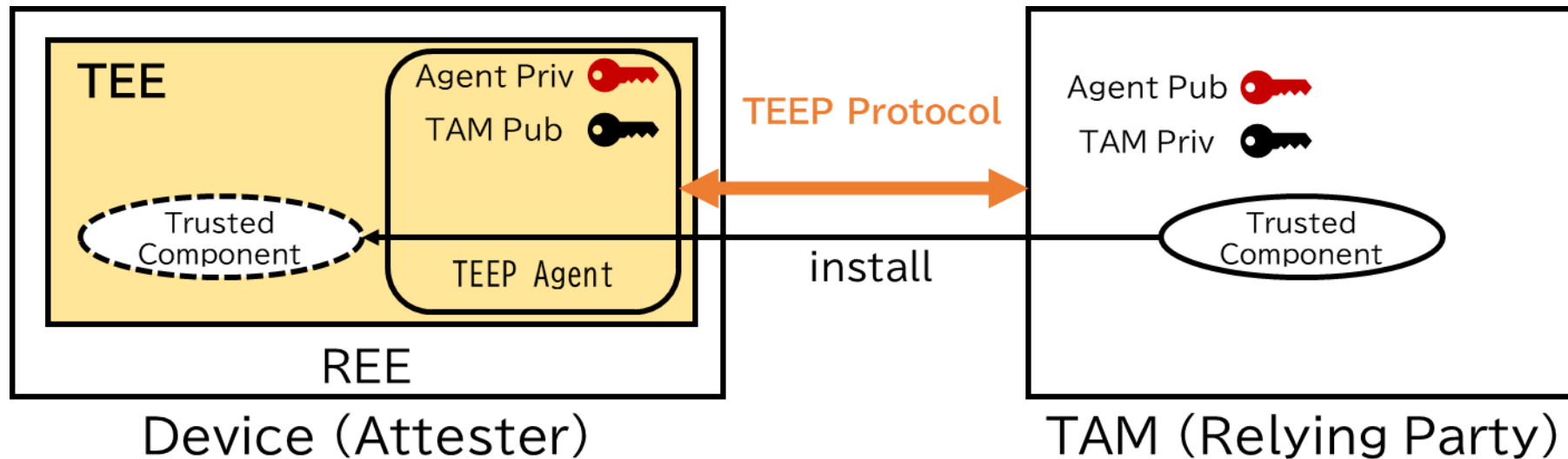
Ken Takayama, Shin'ichi Miyazawa,
Kazuki Zenba and Yuma Nishihira

What this talk is about

- Lessons on designing Evidence for RA-enabled protocols
 - From designing a TEEP implementation with RA on real hardware
 - From comparing LAMPS and SEAT documents
- Key message for protocol designers: Consider both
 - How Evidence is composed and
 - How the bound data is transported

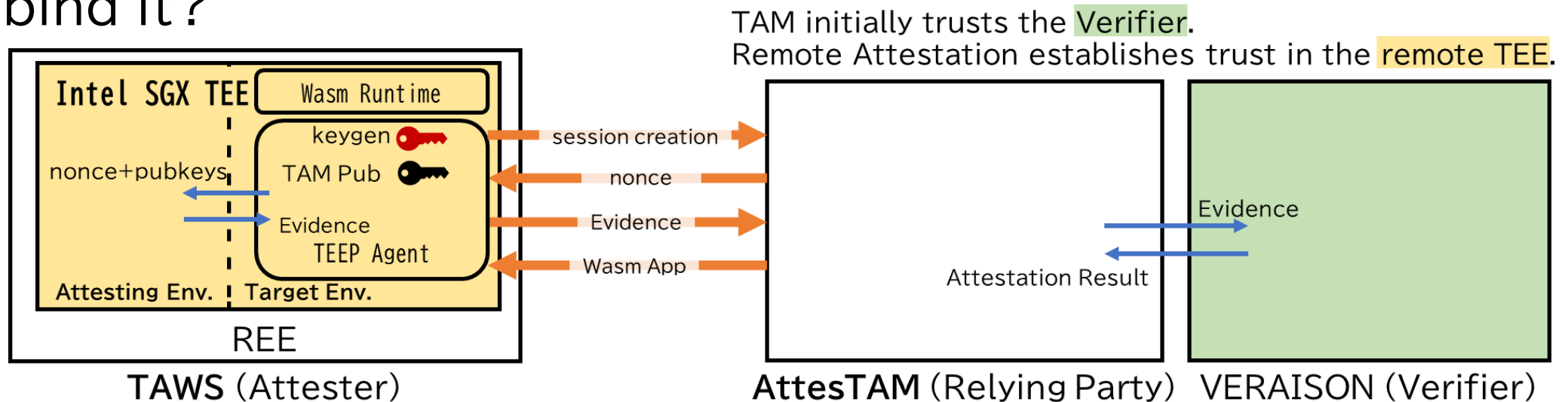
What is TEEP Protocol?

- TEEP provisions Trusted Components into a remote TEE
 - TAM manages the Device and installs programs
- Mutual authentication is required
 - But key establishment is not specified
- Remote Attestation is optional
 - Freshness mechanisms and Evidence composition are not fixed



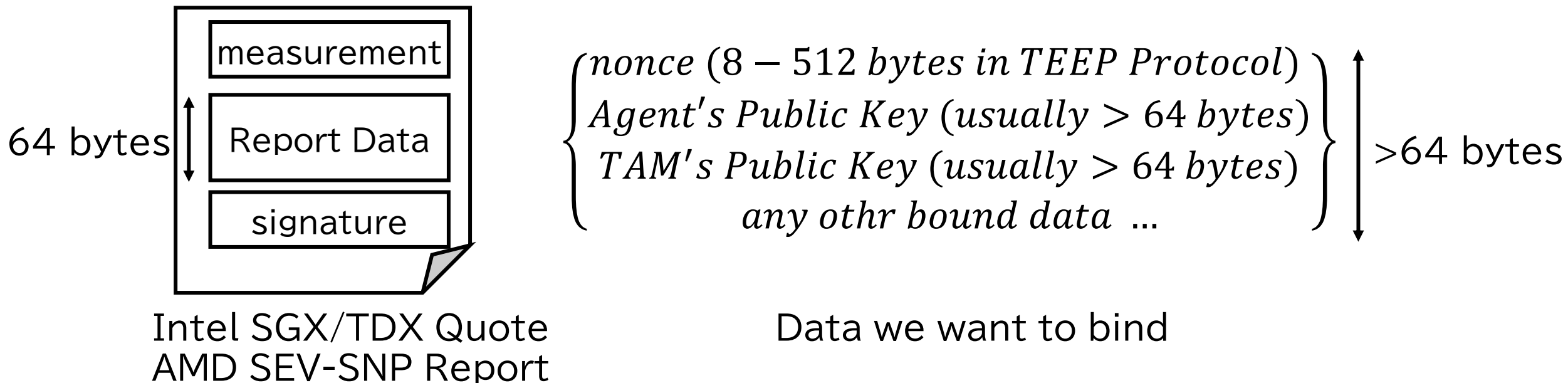
What we implemented at the hackathon

- Goal
 - Establish trust in the TEEP Agent public key via RA
- How we achieve it
 - Generate key inside the TEE
 - Bind it to hardware Evidence with a nonce
- Question (Design challenge)
 - How to bind it?



Evidence format constraints in real TEEs

- Measurement of the TEE state is available
 - Including the TEEP Agent
- Only 64 bytes of Report Data are freely usable
- Binding typically requires more than 64 bytes
 - A nonce + public keys usually does not fit



How existing work addresses this

- Extensible vs Explicit binding:
 - With explicit binding, protocol designers have already addressed it
 - With an extensible protocol, implementers may have to do it

Protocol	Conceptual Bound Context [<i>Terminology</i>]	Evidence Transport
TEEP Protocol	(out of scope)	TEEP message options
CSR Attestation	(out of scope)	CMP, etc.
Key Attestation for EAT	My PubKey + Your nonce + etc.	(out of scope)
SEAT Early Attestation	Our PubKeys + Our nonces + etc. <i>attestation binder</i>	(TODO)
SEAT FACTS	Our PubKeys + Our nonces + etc. <i>Evidence Binding</i>	TLS handshake
SEAT Expat	Our Session + Our nonces + etc. <i>Attestation Binder</i>	Exported Authenticator

Takeaways for protocol designers

- Evidence design has many considerations
 - What to bind (public keys, nonces, measurements, etc.)
 - How to construct it (e.g. EAT, concat + hash extension, etc.)
 - How to bind it with real Evidence format (e.g. Report Data)
 - How to transport it
- Hardware Evidence alone is often insufficient
- Binding and transport strategy choices
 - When flexibility is required: use extensible and structured formats
 - Otherwise: fix the binding scheme and transport to help implementers (e.g. SEAT documents)

Thank you!

- Our project's outcomes:

- [TEEP] RATS+SUIT-integrated reference implementations
- [RATS] CBOR in/out support for VERAISON/{eat, ear}
- [SUIT] SUIT Manifest Processor & Reporting Engine implementation
- [COSE] Fully-specified Algorithms (e.g. ESP256) are supported
 - by python-cwt, t_cose, VERAISON/go-cose

You can try our hands-on demo

(TAM + TEEP Agent in Intel SGX simulation + VERAISON)

<https://github.com/s-miyazawa/teep-wasm-demo>

This work was supported by JST K Program Grant Number JPMJKP24U4, Japan.