Direct Anonymous Attestation and Identity Management

Stefanos Vasileiadis
Security Engineer, Security Group

UBITECH Ltd

ASSURED webinar on Attestation Capabilities

webinar | 31 May 2023

Direct-Anonymous-Attestation (DAA)

Direct Anonymous Attestation is a special kind of group signatures, that uses blinded credentials, authorized by a Trusted certification authority, in order ,for the Enrolled device, to perform anonymously signing.

Main Research Topics

- Enhanced DAA with revocation capabilities
- Attribute Based DAA

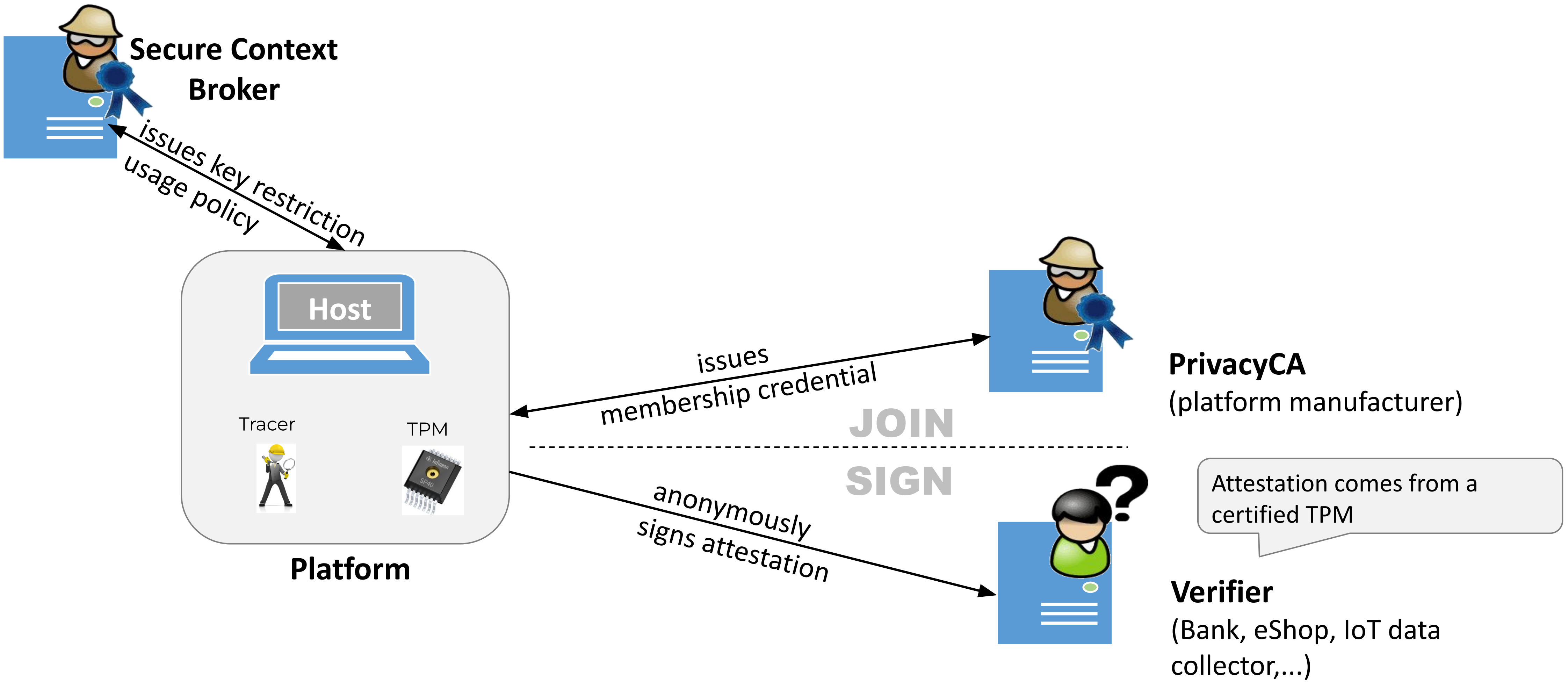
DAA-Security Properties

➔ **User-Control Anonymity**
  - ◆ The identity of the user cannot be revealed and multiple signatures cannot be linked.

➔ **Non-Frameability**
  - ◆ Any Adversary should not be able to impersonate honest platforms.

➔ **Unforgeability**
  - ◆ Valid signatures are only producible by honest platform and are verifiable & linkable when specified.

Trusted Computing Base



Direct Anonymous Attestation Break Down

DAA Phases

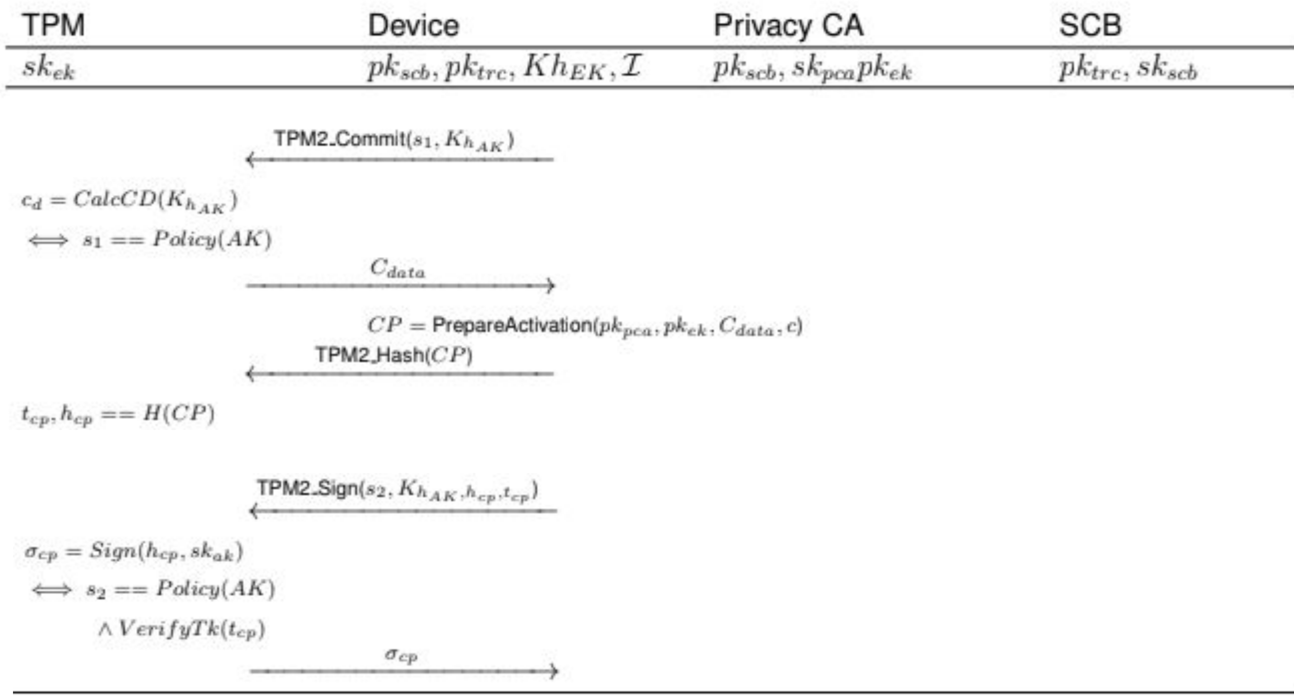
- Join Phase
- Sign Phase

DAA Join Phase

- The Device sends the TPM's Endorsement Key to the privacyCA.
- The privacyCA creates a fresh challenge binded to this TPM.
- The Device requests from the TPM to open the challenge.
- The Device sends to the privacyCA the opened challenge.
- The privacyCA verifies the Device's response.
- The privacyCA creates the DAA credentials, which are the encrypted with a symmetric key specific to this Device.
- The Device requests from the TPM to extract the symmetric key and decrypt the DAA credential.
- The DAA credential are then stored for future use.

DAA Sign

- The Device requests from the TPM to create a DAA signature.
- The Device uses the DAA credential acquired by the end of the join phase to finalize the DAA signature.
- The DAA signature is sent to the privacyCA.
- The privacyCA tries to verify the DAA signature.
- \+ If the verification is successful the Device can use its DAA key

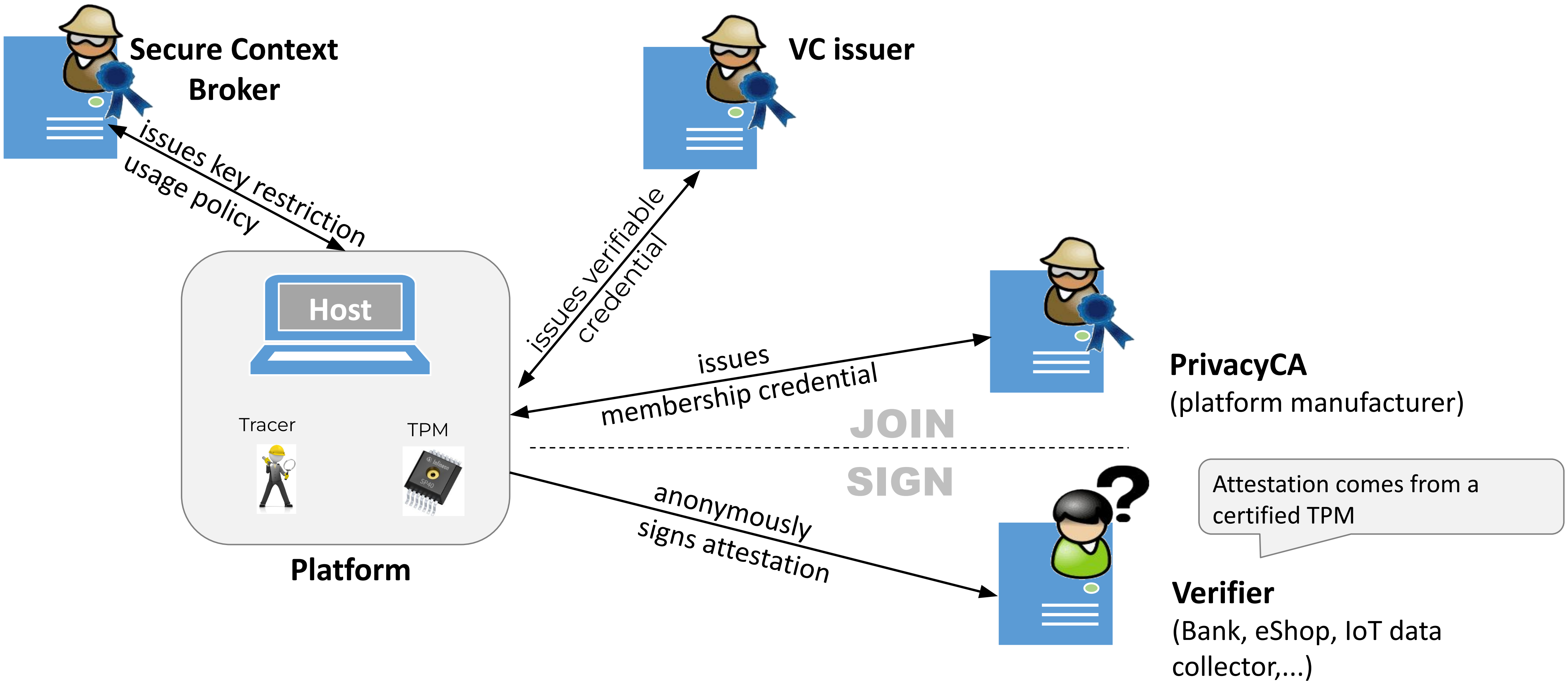


DAA Evaluation

Secure Enrollment Phase	Command	Average [s]	Min [s]	Max [s]
Initialization of JOIN phase	TPM2_CreatePrimary	0.3008816104	0.299938289	0.302655462
Creation of authorization ticket	TPM2_LoadExternal	0.2257807849	0.223496727	0.226901812
	TPM2_VerifySignature	0.3021180071	0.299876338	0.303544441
	TPM2_FlushContext	0.1623935827	0.160487621	0.16396455
Activate credential	TPM2_ActivateCredential	0.3498275717	0.34769205	0.352471702
Satisfy policy for commit	TPM2_StartAuthSession	0.1601472872	0.157956615	0.161473438
	TPM2_policyCC	0.1827594241	0.180977454	0.184460488
	TPM2_policyOR	0.1833791188	0.180766649	0.184615092
	TPM2_policyAuthorize	0.1938004759	0.19287893	0.194812401
	TPM2_Commit	0.2375329302	0.23554051	0.23879233
	TPM2_Hash	0.13857767	0.137529965	0.139764075
Satisfy policy for signing	TPM2_StartAuthSession	0.1597817141	0.157091723	0.160574503
	TPM2_policyPCR	0.1873876674	0.185655464	0.188756125
	TPM2_policyOR	0.4039362717	0.39865777	0.435070658
	TPM2_policyAuthorize	0.188273368	0.186415087	0.190004208
	TPM2_Sign	0.3408976302	0.336817956	0.349208023

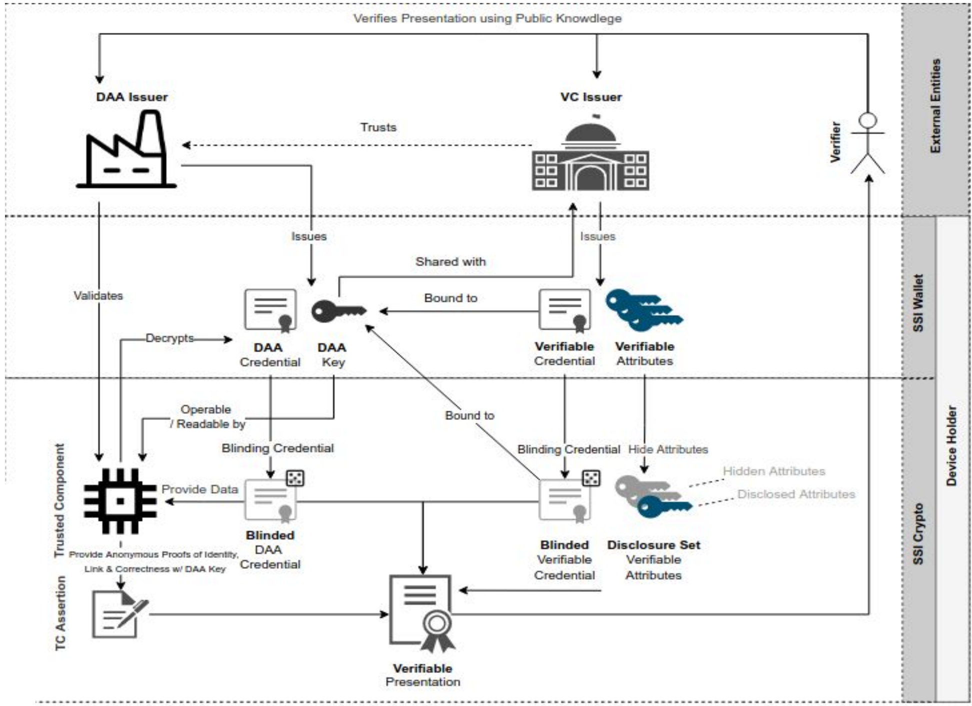
TPM-Based Wallet

Trusted Computing Base



TPM Wallet Properties

Holder Binding	It must be ensured that the issued identity data are delivered only to the intended Holder.
Device Binding	Issued VCs should be bound to the Holder's unique identifier ,the DAA key.
Selective Disclosure	VPs should constitute collections of claims that the Holder can construct disclosing only those attributes needed for verification without revealing further information.



Full Anonymity

Unforgeability

Wallet Correctness

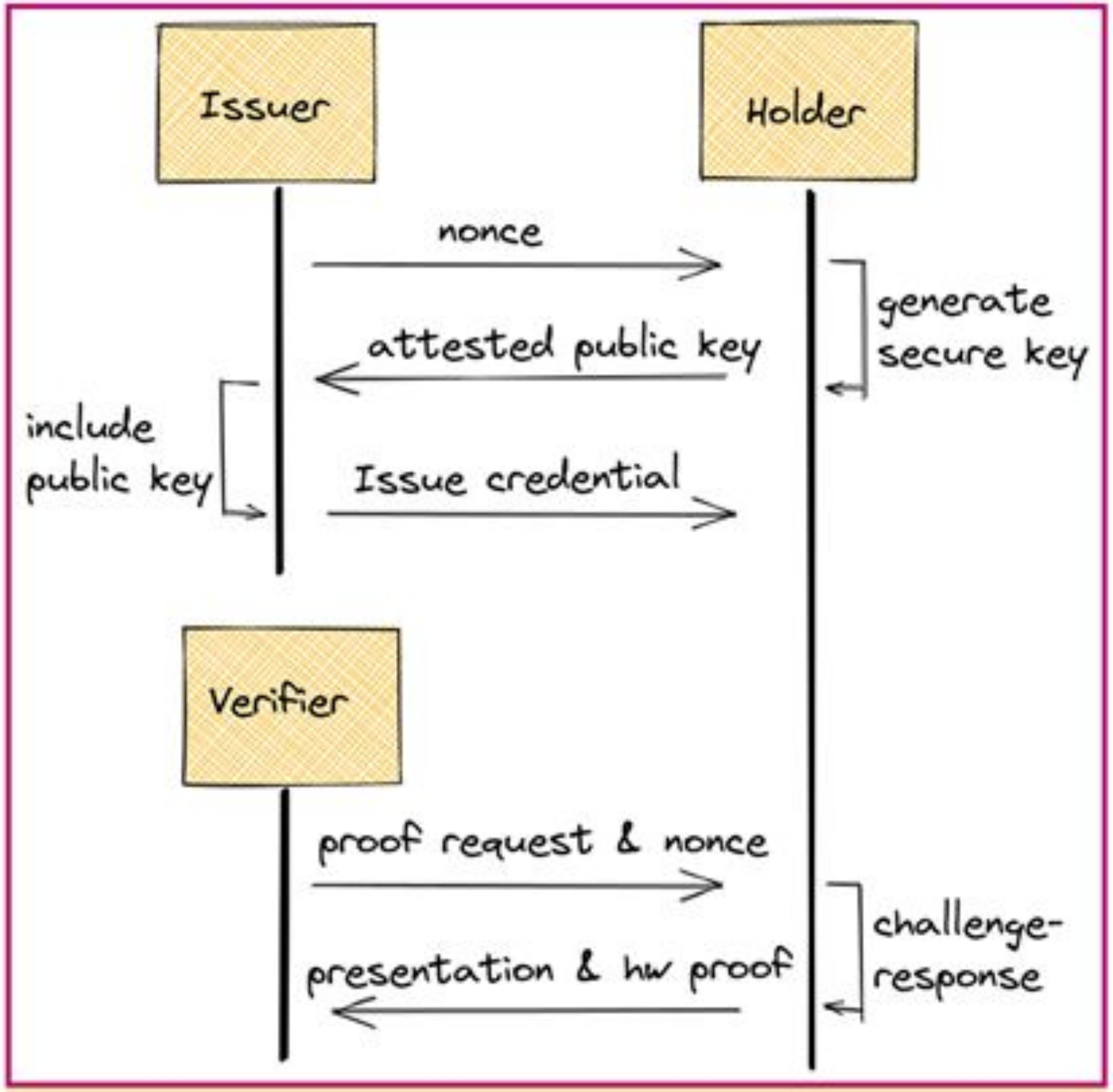
DAA TPM WALLET
Break Down

Phases

- Join Phase
- Get Verifiable Credential Phase
- Create Verifiable Presentations

Get Verifiable Credential Phase

- The Device sends to the VC Issuer the DAA public key along with a DAA signature as proof of correctness of the Device.
- The VC Issuer generates the relevant attribute keys for the VCs
- Includes the DAA public key as an identity attribute.
- The Device receives the VCs and store them for future use.



Create Verifiable Presentations

- The Device blinds the Verifiable Credential (VC) with random numbers to assure unlinkability.
- The Holder chooses which attributes wants to disclose the are others remain hidden.
- The Device requests from the TPM to create a DAA signature.
- The Device use the DAA signature and the blinded VC , disclosing only the desired attributes, to finalize the Verifiable Presentations.

$TPM\ (x_0)$		$Wallet\ (PK, Key_W)$
		$a \in_R \mathbb{Z}_q.$ $A' = aA, B' = aB, C' = aC\ D' = aD$ $A'_w = aA_w; B'_w = aB_w$ $C'_w = aC_w, D'_w = aD_w$ $E'_{W_k} = aE_{W_k}\ \forall\ k \in [0, n]$
	$\xleftarrow{\texttt{TPM2\_Commit}(B' + E'_{W_0})}$	
$\omega_0 \in_R \mathbb{Z}_q$ $R_0 = \omega_0(B' + E'_{W_0})$		
	$\xrightarrow{R_0}$	
		$\{\omega_1, \ldots, \omega_p\} \in_R \mathbb{Z}_q$ $R_{W_k} = \omega_k E'_{W_k}\ \forall\ k \in \mathcal{P}$ $c = H_1(A' \vert B' \vert C' \vert D' \vert A'_w \vert B'_w \vert C'_w \vert D'_w \vert E'_{W_0} \vert E'_{W_1} \vert \ldots \vert E'_{W_n} \vert R_0 + \sum_{k \in \mathcal{P}} R_{W_k} \vert m')$
	$\xleftarrow{\texttt{TPM2\_Sign}(c)}$	
$s_0 = \omega_0 + cx_0$	$\xrightarrow{s_0}$	
		$s_k = \omega_k + cx_k\ \forall\ k \in \mathcal{P}$ $\sigma = (A', B', C', D', A'_w, B'_w C'_w, D'_w, E'_{W_0}, \ldots, E'_{W_n}, s_k, s_0, c)$

DAA TPM Wallet Evaluation

Activity	Mean	± (95% CI)
Host Calculations	33.63 ms	4.04 ms
Total TPM Time	1324.36 ms	44.80 ms
TPM2_StartAuthSession	52.48 ms	2.67 ms
TPM2_PolicyCommandCode	1.51 ms	0.03 ms
TPM2_PolicyOR	3.11 ms	0.09 ms
TPM2_PolicyAuthorizeNV	326.28 ms	7.41 ms
TPM2_Commit	176.63 ms	3.23 ms
TPM2_Hash	119.27 ms	9.41 ms
TPM2_StartAuthSession	51.32 ms	2.64 ms
TPM2_PolicyPCR	2.57 ms	0.07 ms
TPM2_PolicySigned	141.02 ms	2.06 ms
TPM2_PolicyOR	3.18 ms	0.10 ms
TPM2_PolicyAuthorizeNV	325.37 ms	7.74 ms
TPM2_Sign	121.62 ms	9.35 ms
Total Create Time	1357.99 ms	48.84 ms
Verify Presentation	85.40 ms	5.02 ms

Application Domains & Road Ahead

APPLICATIO
N DOMAINS

Connected Cars (CCAM)

Zero Trust Concept: Dynamic trust assessment based on which involved entities can establish trust for collaboratively executing functions

Chip-to-cloud assurance for edge and cloud working in tandem

<u>TC component to be added in each heads ECU</u>

Smart Cities

✓ **Anonymity and privacy of users**
✓ Strong **authentication and authorization**
Attribute-based Access Control
✓ **Decentralized Identity Management**
✓ Integration of legacy devices
✓ Attestation

Smart Aerospace

Move towards **remote maintenance**

Strong guarantees on **secure software updates**

Efficient certification and fast auditing

<u>Re-configuration of hardware</u> for security

RISC-V Architectures

4-layered security sandbox for safeguarding the entire lifecycle of devices

Formally verified security controls comprising specific instruction sets

Efficient attestation of only those updates that break the trust model

ASSURE
CONNECT
ccam trust & resilience
REWIRE
DOOR
ESSIF - LAB
SUBGRANT

THANKS

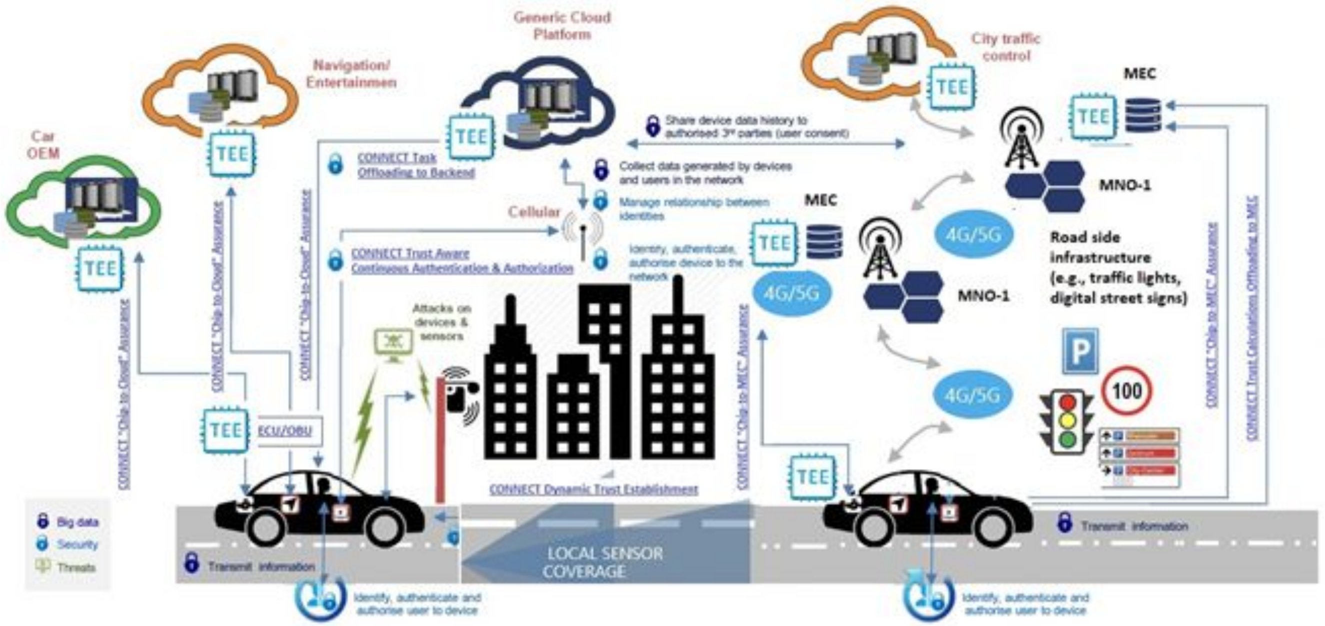
PROJECT-ASSURED.EU

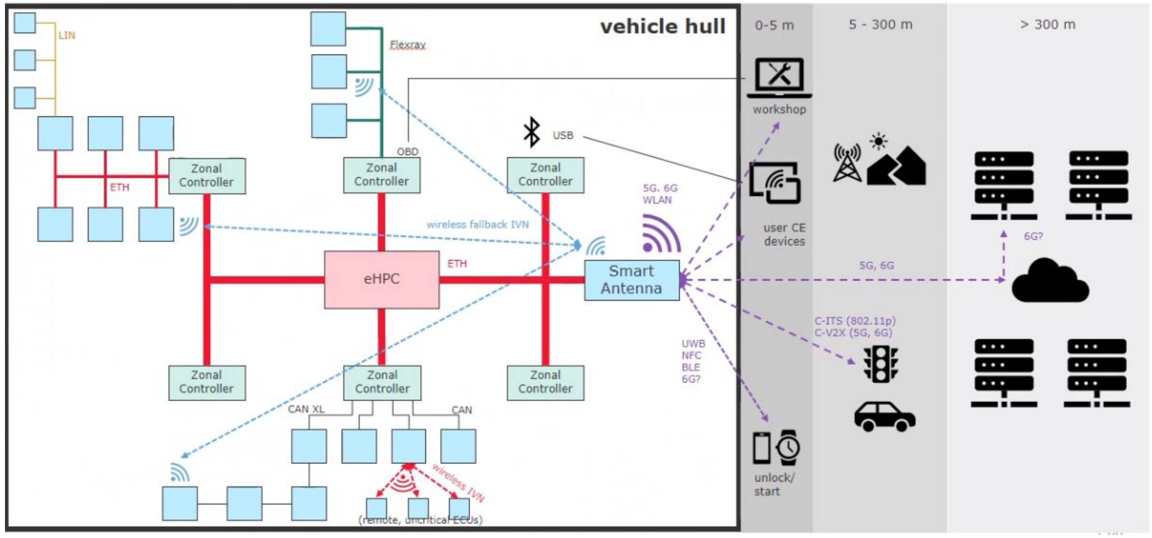
@Project_Assured

ASSURED project is funded by the EU's Horizon2020
programme under Grant Agreement number 952697

Trust Management for Resilient CCAM

- Secure “Chip-to-cloud” assurance solutions for enabling trustworthy and resilient safety-critical services
- Convergence of security and safety
- **Interlinking attestation and secure offloading mechanisms**
  - ✓ *TC-enabled middleware that simplifies the trust relationships between all layers in edge-cloud runtime stack*
- Verifiable Presentations as assertions
  - ✓ Securely (cryptographically) vehicle-issued claims on the device attributes
  - ✓ Mapped to different levels of trust based on the mixed-criticality of the target services
- Scenarios of Interest
  - ✓ ***Vehicle’s Cooperative Situation Awareness*** *(FIAT)*
  - ✓ **In-vehicle application relocation and software migration** (DENSO)
  - ✓ **Misbehavior Detection** (IRTSX)





Cloud-Edge-End to realize ubiquitous computing brings forth new challenges

Moving forward - Still a number of challenges

SECURITY, PRIVACY, TRUSTWORTHINESS

- **Distributed:** Next-generation systems must be seen as inherently and increasingly Federated Safety Critical Systems that are not owned by a single entity
- **Bottom Up:** Data and system components must be in position to make strong statements about their (runtime) integrity
- **Sustainable Security?**
- **Quantum Secure TPMs**

SAFETY

Interdependable requirements are characterized as being complex, multi-dimensional, transversal, uncertain and with different and potentially contradicting requirements

Interdependable Runtime Solutions for Next-Generation Smart Systems

Key Restriction Usage Policies

In order to enhance the DAA protocol we exploit the key restriction usage policies.A Trusted Third Party (SCB) constructs a policy digest, which will be binded with the DAA key.

- Load from the Platform Configuration Registers a pre-defined state of the Device (TPM2_policyPCR).
- Verify the the freshness and integrity of the the output of the Tracer (TPM2_policyOR | TPM2_policySIGNED).
- Use a certificate from a Trusted Third Party (SCB) to construct the final run-time policy Digest and gain access to the DAA key (TPM2_policyAUTHORIZE).
  - Using TPM2_policyAUTHORIZE the protocol can support updates during run-time.

Initiate Revocation Index

- Host builds a PolicySecret policy based on the ACI name
- Creation of Attestation Key
- To create the index, the host calculates a new policy digest that links the index's usage to the AK by leveraging the TPM2_policyAuthorize.
- By issuing the DefineSpace command, the index is now built within the TPM.

Initialize revocation index: TC $\rightleftharpoons$ HOST

$ACI, r_{index}, A_{\mathtt{TMP}}, AK_{name}$

$P_d := \mathtt{hash}(CC_{PolicySecret} \,||\, ACI.\mathtt{name})$

$A_{\mathtt{TMP}}.\mathtt{Policy} := P_d$

$\xleftarrow{\mathtt{CreatePrimary}(A_{\mathtt{TMP}}, \mathtt{Owner})}$

$D := \mathtt{KDF}(\mathtt{HierarchySeed}(\mathtt{Owner}), A_{\mathtt{TMP}})$

$KH_{AK}, AK := \mathtt{CreateKey}(D)$

$\xrightarrow{KH_{AK}, AK_{pub}}$

$P_d := \mathtt{hash}(CC_{PolicyAuthorize} \,||\, AK_{name})$

$\xleftarrow{\mathtt{DefineSpace}(P_d, r_{index})}$

$\mathtt{CreateSpace}(r_{index}, P_d)$

$\Longleftrightarrow \mathtt{SpaceNotDefined}(r_{index})$

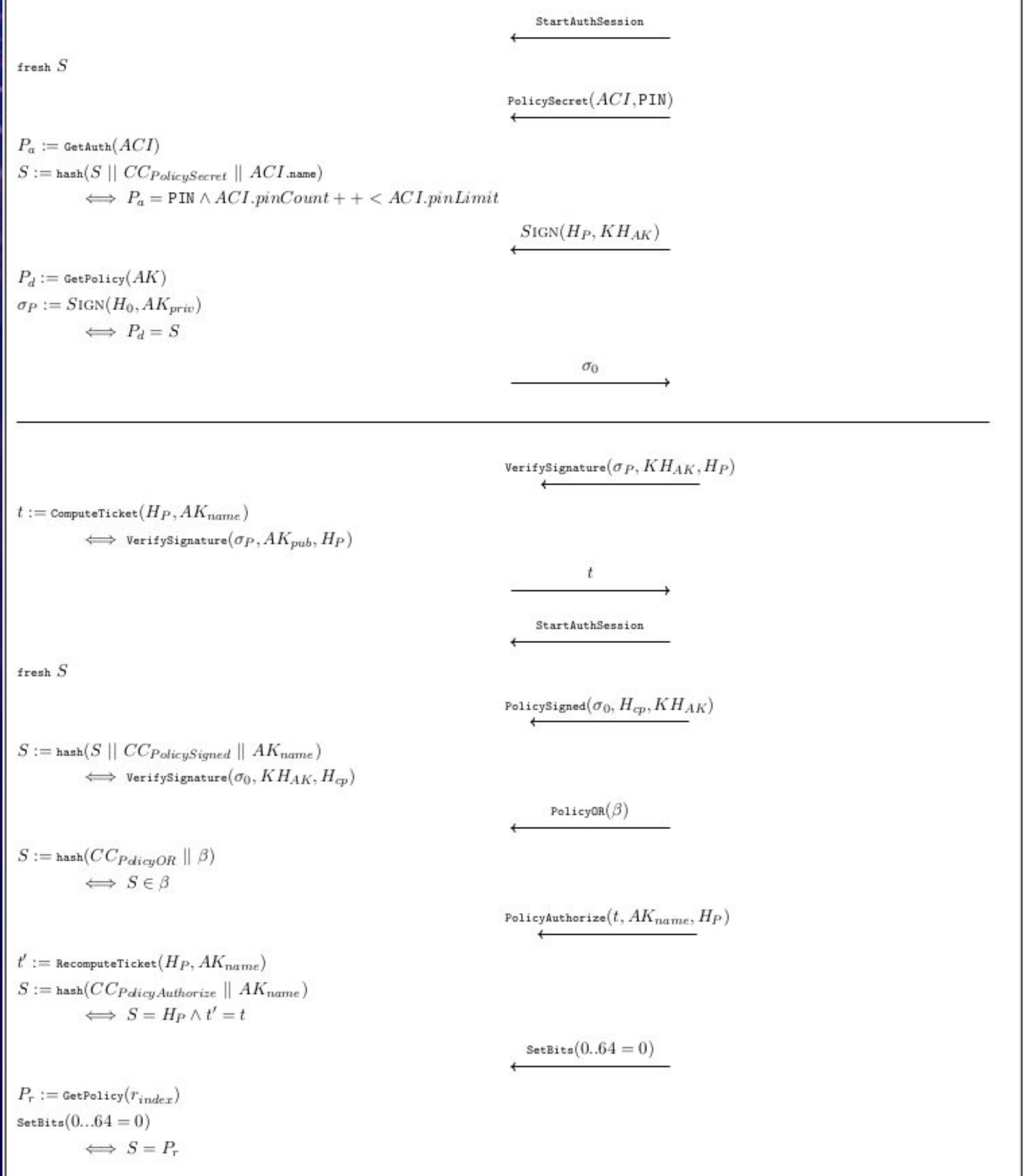
Activate Revocation Index

- Hashing the policy and generating the command parameter hash for the initial write.
- Gains access to the AK by providing the ACI's secret, inherently incrementing the authorization count.
- Now the host can get a signature over the policy and acquire a verification ticket.
- Host initiates a new session .
- Executes one of the index's valid policies, namely the TPM2_policySigned with the previously acquired signature.

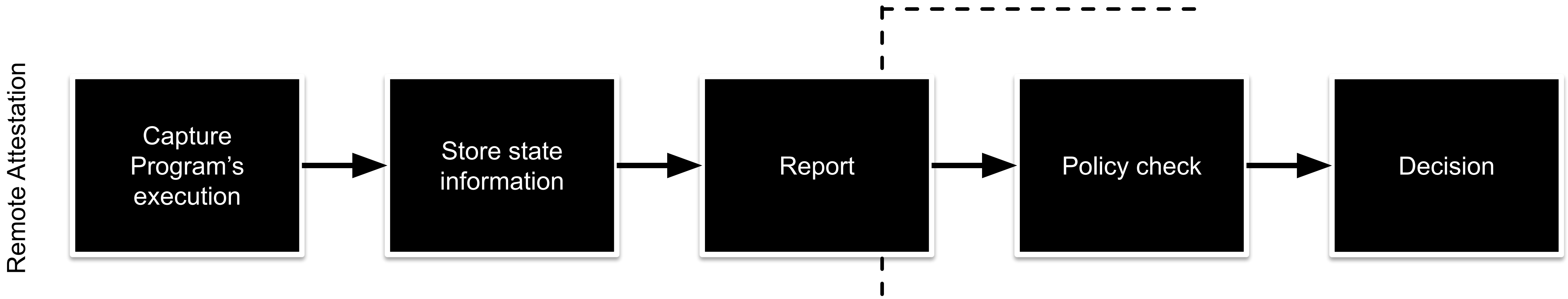
Provision revocation index: TC	$\rightleftharpoons$	HOST
		$ACI, \mathtt{PIN}, A_{\mathtt{TMP}}, \mathcal{P}, \beta, r_{index}$
	$\xleftarrow{\mathtt{CreatePrimary}(A_{\mathtt{TMP}}, \mathtt{Owner})}$	
$D := \mathtt{KDF}(\mathtt{HierarchySeed}(\mathtt{Owner}), A_{\mathtt{TMP}})$ $KH_{AK}, AK := \mathtt{CreateKey}(D)$		
	$\xrightarrow{KH_{AK}, AK_{pub}}$	
		$H_P := \mathtt{hash}(\mathcal{P})$ $H_{cp} := \mathtt{hash}(CC_{NV\_SetBits} \mid\mid r_{index}.name \mid\mid 0, 0..0)$ $H_0 := \mathtt{hash}(TC_{data} \mid\mid H_{cp})$
	$\xleftarrow{\mathtt{StartAuthSession}}$	
$\mathtt{fresh}\ S$		
	$\xleftarrow{\mathtt{PolicySecret}(ACI, \mathtt{PIN})}$	
$P_a := \mathtt{GetAuth}(ACI)$ $S := \mathtt{hash}(S \mid\mid CC_{PolicySecret} \mid\mid ACI.\mathtt{name})$ $\iff P_a = \mathtt{PIN} \land ACI.pinCount{+}{+} < ACI.pinLimit$		
	$\xleftarrow{\textsc{Sign}(H_P, KH_{AK})}$	
$P_d := \mathtt{GetPolicy}(AK)$ $\sigma_P := \textsc{Sign}(H_P, AK_{priv})$ $\iff P_d = S$		
	$\xrightarrow{P_{\mathtt{Auth}}}$	

Activate Revocation Index (2)

- The current session digest should now match the branch digest b0, and the host executes TPM2_policyOR with β.
- After a successful verification from the TPM, it will replace the session digest with a concatenation of all provided branch digests in β.
- The Host executes TPM2_policyAuthorize with the previously acquired ticket and signature. The TPM then verifies the ticket, the signature, and finally, the session digest
- We can now execute a write operation (through the SetBits command), and the index has been activated



Device Operational Assurance – Remote Attestation



- Quite a lot of work has been done in the context of Remote Attestation for verifying the correct state of a device
  - ✔ "State" can include binary configuration integrity, secure boot, etc.
  - ✔ **TPM Fundamentals**
  - ✔ But the pressing question is how to check also properties of interest during runtime for attesting various Levels of Assurance (LoAs)

Keys

Can be linked to *policies*, that can make the key usable under certain conditions, or limit the abilities of the key.
Can *not* be moved to another platform*.

PCRs

Contains a digital fingerprint of the residing platform. Firmware, bootloader, etc. Can even hold fingerprints of applications.

Adding HW-based keys to the Wallet

By adding a **unique TPM key** to every issued **credential**, the issuer can get guarantees that the credential is safe. How?

By signing all **presentations** with this key. But it requires some properties.

KEY PROPERTIES

- May only be used in a Trusted State (PCRs)
- May only be used by authorization of the wallet
- Can only be used on particular TPM (hence: platform)
- Can provide *unlinkable* signatures (DAA)
- Policy (Trusted State) can be updated by the issuer, and only the Issuer.

All **presentations** must be signed by this key. If verified, the verifier knows

- The presentation comes from a trusted platform according to the issuer - if the verifier trusts the issuer, it now trusts that the presentation isn't a product of malicious software
- The presentation is made through an authorized wallet
- The credential has not been moved … Or has it?

USE OF DIRECT ANONYMOUS ATTESTATION (DAA)

It's not possible for the verifier to determine *which* key gave the signature, only that it was valid and issued by that Issuer. What if a *similar* holder provided a signature, but he had a different security level?

SOLUTION

- Make the key part of the credential
  - Only a platform that can load that particular key, can get the claims.
  - This is possible through **Attribute-based Direct Anonymous Attestation (DAA-A)**

This solution is being developed and tested under the eSSIF framework.