SNIA DEVELOPER CONFERENCE

SDC 25

By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

Navigating NVMe-oF Authentication

Best Practices for Key Management in Pre-AVE and Post-AVE Environments

Swati Chawdhary (Senior Manager)

Raj Kumar Dani (Associate Director)

www.sniadeveloper.org

SDC 25

Agenda

- Background
- NVMe-oF Authentication Basics
- DH-HMAC-CHAP Authentication
- Security threats
- Single/Multiple Keys – how it works
- Our experiments
- AVE Basics
- Conclusion

Background — continuation of our SDC talk

Last year (SDC 2024) — recap

- Talk: “Evaluating Discovery Automation on a Large-Scale NVMe-oF Deployment”
  (https://www.snia.org/sniadeveloper/session/18405)
- Shared our experience and challenges deploying automated discovery in a highly scaled NVMe-oF environment
- Focus: discovery flow, scale/ops pain points, lessons from production-like testing

This year (2025) — what’s new

- This session continues that work with **NVMe-oF Discovery Automation with Authentication**
- What’s new: In-band authentication (DH-HMAC-CHAP), key-provisioning, and AVE
- Why it matters: security enhancement & blast-radius vs. deployability at scale
- What we’ll share: Our experience and challenges we faced

NVMe-oF Authentication Terminology

NVMe-oF : Non-Volatile Memory Express over Fabrics

Extends NVMe across a network fabric (TCP/RDMA) so hosts can access NVMe devices as if they were local, preserving the NVMe command set and low-latency behavior

NVMe-oF supports 2 types of authentication methods

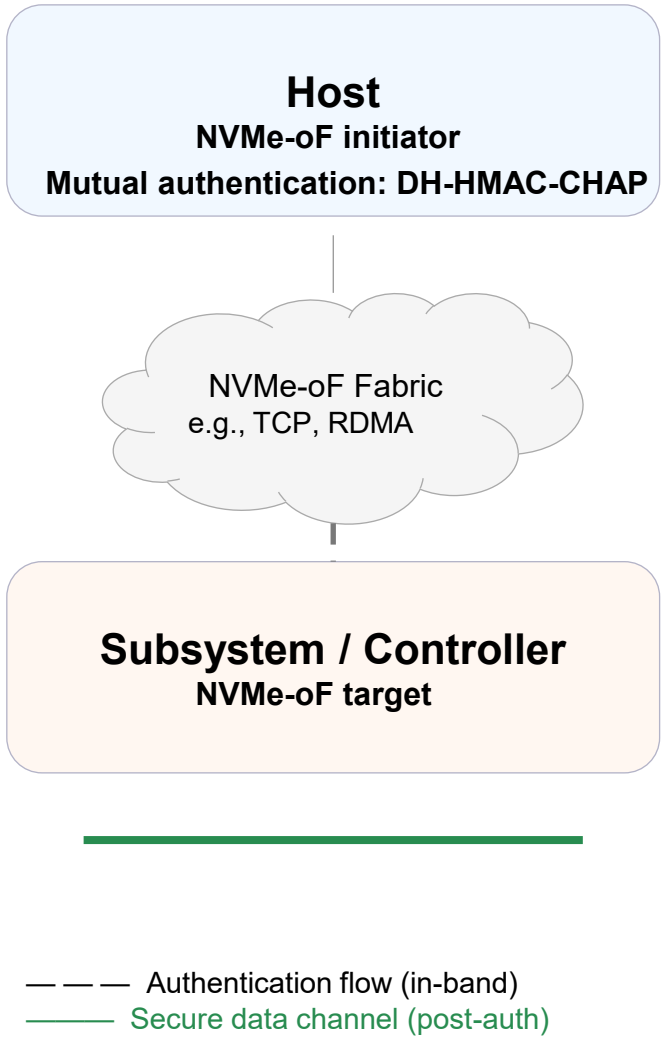
Authentication (in-band: DH-HMAC-CHAP)
Secure Channel (TLS 1.3, NVMe/TCP)

In-Band Authentication Basics

Happens inside NVMe-oF connection setup (no separate control plane)
Challenge–response; Keys are never sent on the wire.
Uni-directional (host→subsystem) or bi-directional (mutual); DH-HMAC-CHAP

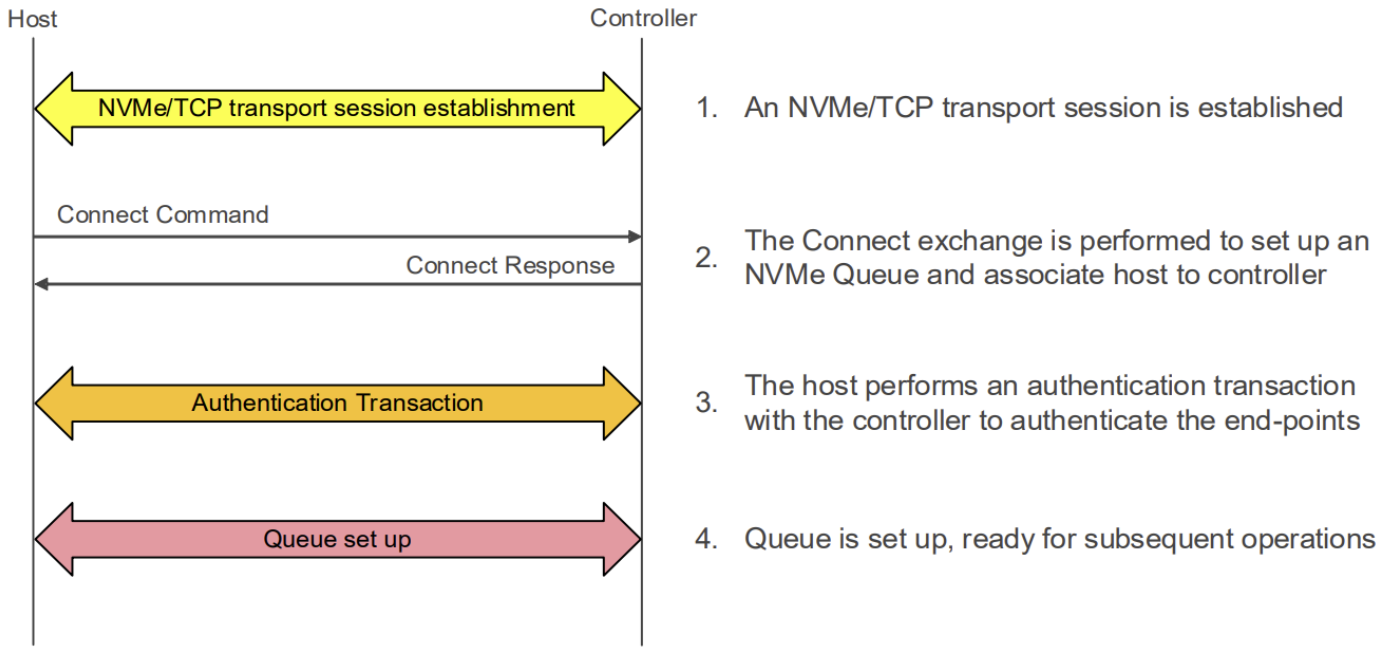
Key terms

Entities: Host, NVM Subsystem/Controller.
Roles: Initiator (Host) and Responder (Controller) during auth.
Keys: K_H (host key), K_S (subsystem key);



NVMe-oF In-Band Authentication on NVMe/TCP

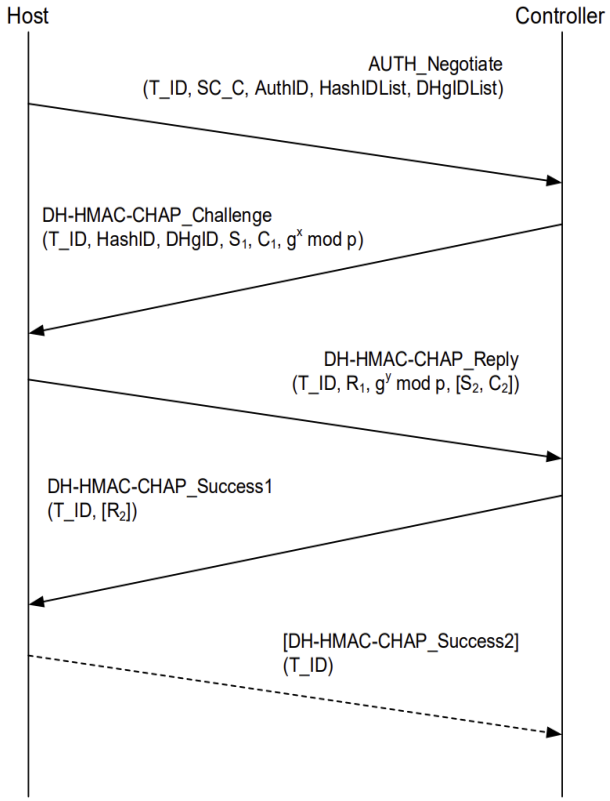
- 1) Establish TCP session
- 2) Connect: set up NVMe queues & associate host ↔ controller
- 3) Authentication transaction (In-Band Authentication; DH-HMAC-CHAP)
- 4) Queue ready for subsequent operations



- Pre-requisite: Authentication enabled on both Host and Controller(keys/provisioning)

DH-HMAC-CHAP In-Band Authentication

- Challenge-response proof of key knowledge — **keys never sent on wire**
- Host proves it knows K_H
- Controller proves it knows K_C
- Core formula: R = HMAC(K_responder, C || context)



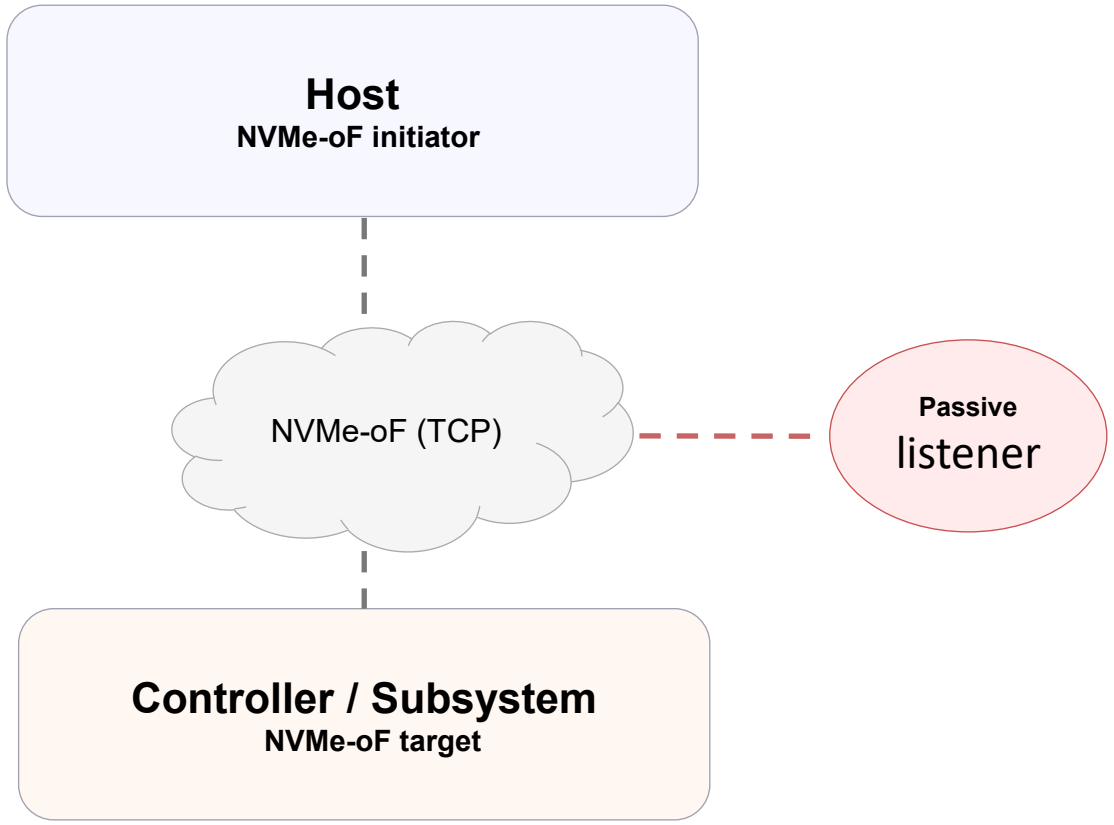
- AUTH_Negotiate: agree on hash & DH group
- Challenge: Controller → Host (includes C1, DH params)
- Reply: Host → Controller (R1 + optional S2,C2)
- Success1/Success2: mutual verification & completion

Security threats

In-Band Authentication

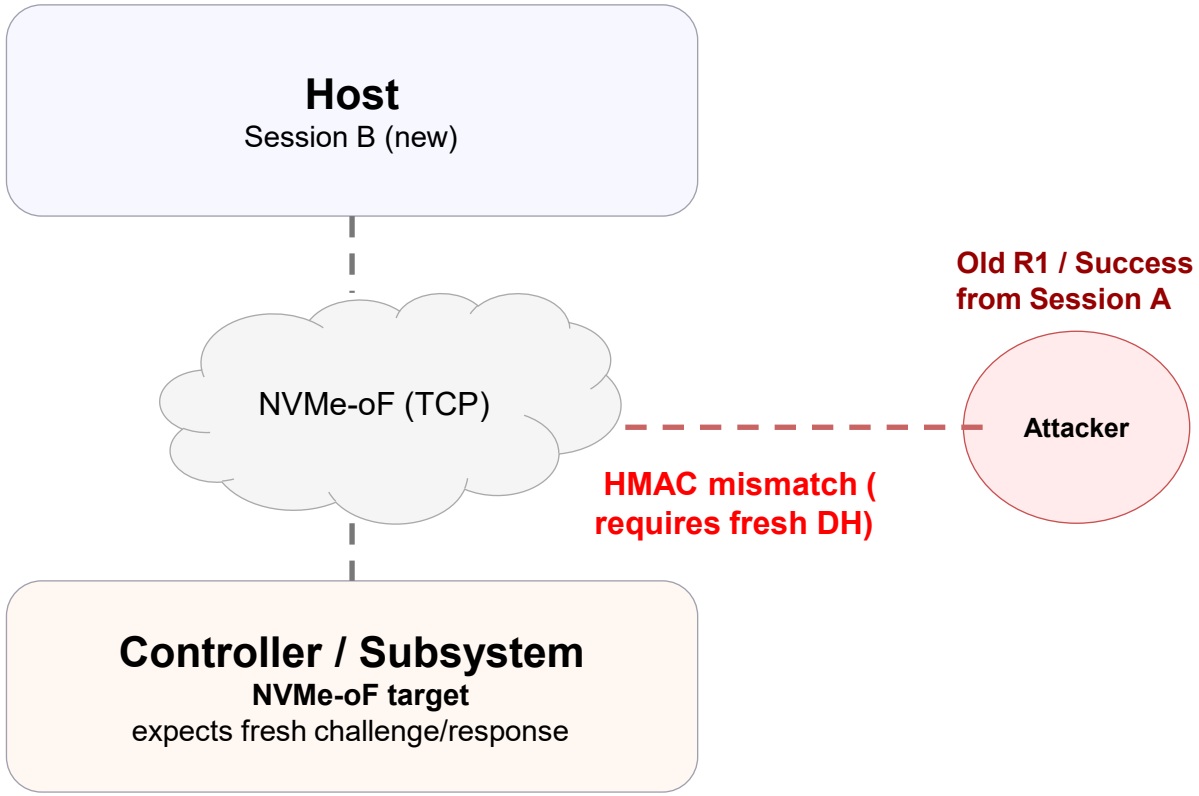
Passive eavesdropping — listener on the wire

- **Passive eavesdropping**
  - An attacker only listens on the wire
  - Trying to learn Keys or harvest messages for later use
- **How DH-HMAC-CHAP solves it**
  - An attacker only sees: DH publics, HMAC tags — not the long-term secret.



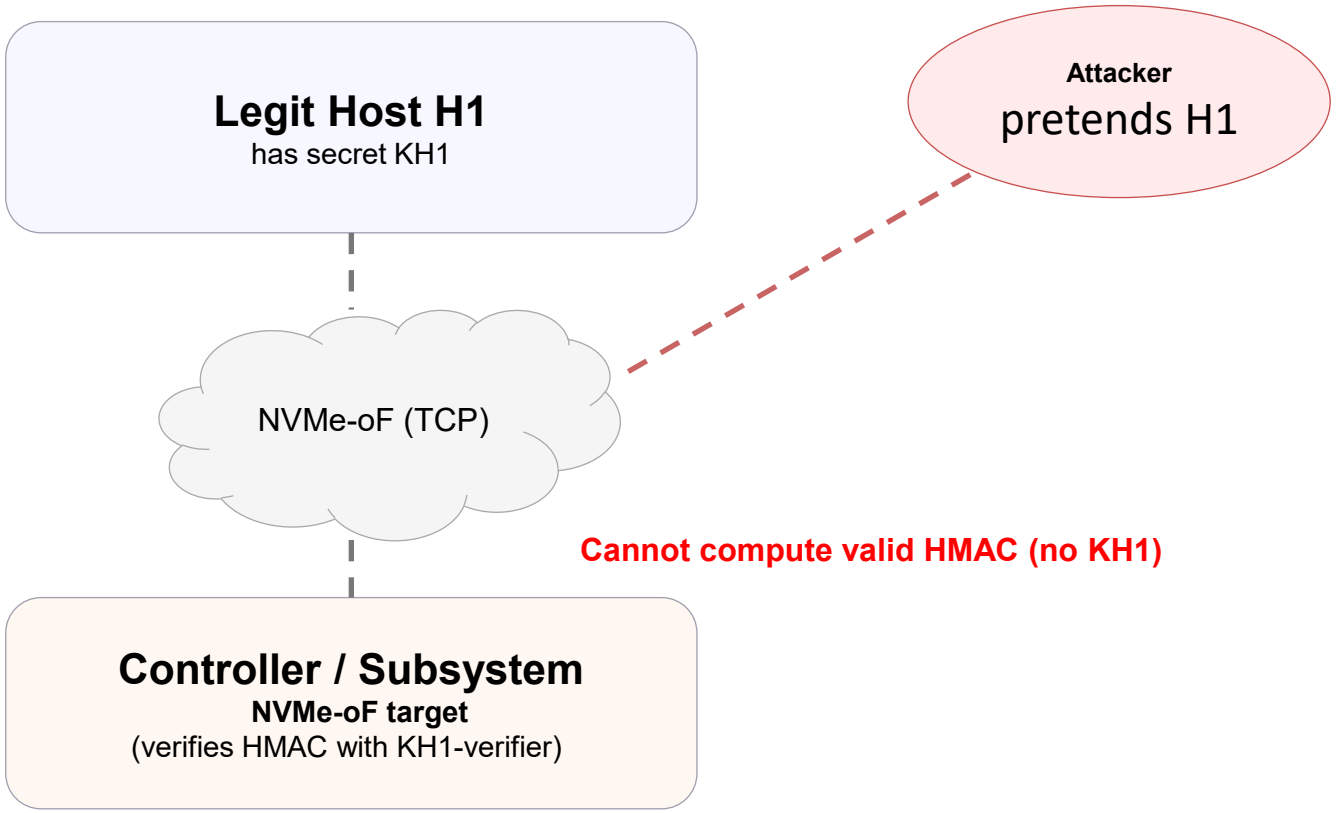
Replay — reusing captured auth messages

- **Replay**
  - An attacker reuses a previously captured authentication message(Old R1/Success)
  - Attacker tricks a new session with previously captured authentication messages
- **How DH-HMAC-CHAP solves it**
  - As all messages and id's are valid only for that session
  - HMAC calculation mismatch (requires fresh DH)



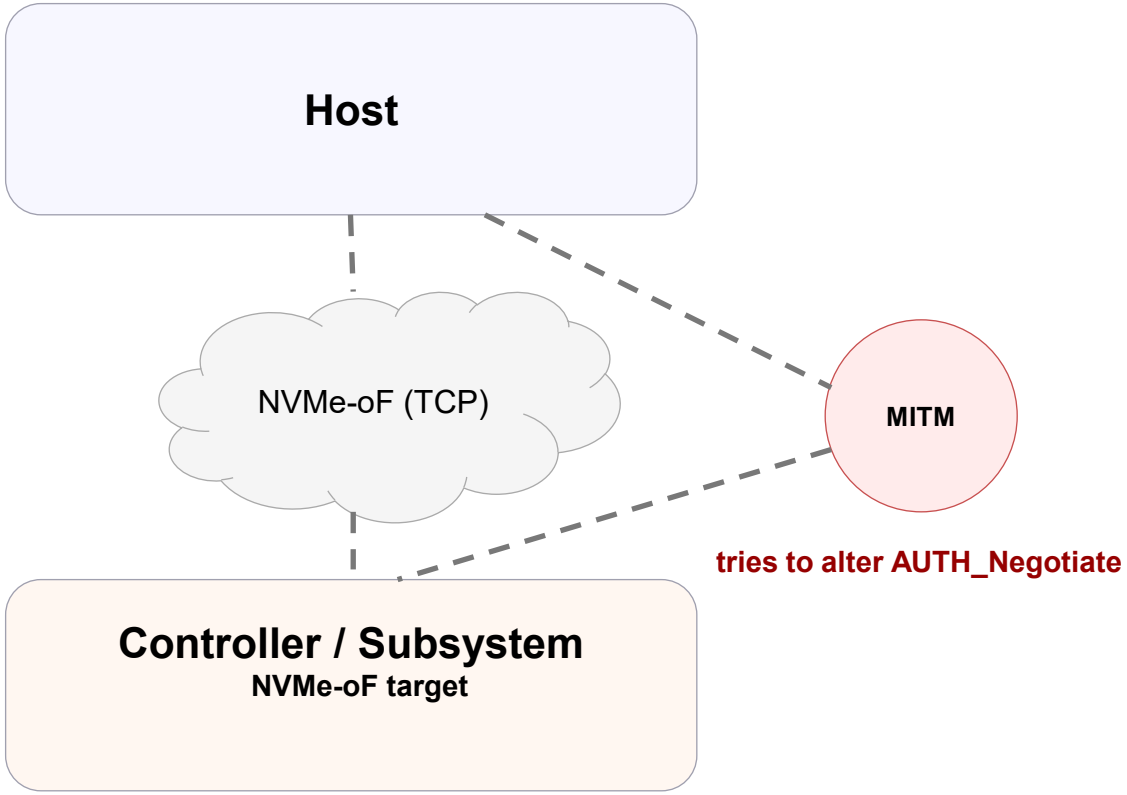
Impersonation — fake host or fake target

- **Impersonation**
  - An attacker pretends to be a valid host or target to gain access without owning the key
- **How DH-HMAC-CHAP solves it**
  - An attacker cannot compute valid HMAC
  - The attacker doesn’t know the valid key



Man-in-the-Middle — tamper/downgrade attempts

- **Man-in-the-Middle**
  - An attacker sits between host and controller
  - Relaying/altering messages or forcing weaker settings.
- **How DH-HMAC-CHAP solves it**
  - HMAC binds IDs + fresh nonces + DH
  - Tamper with any of these, breaks verification

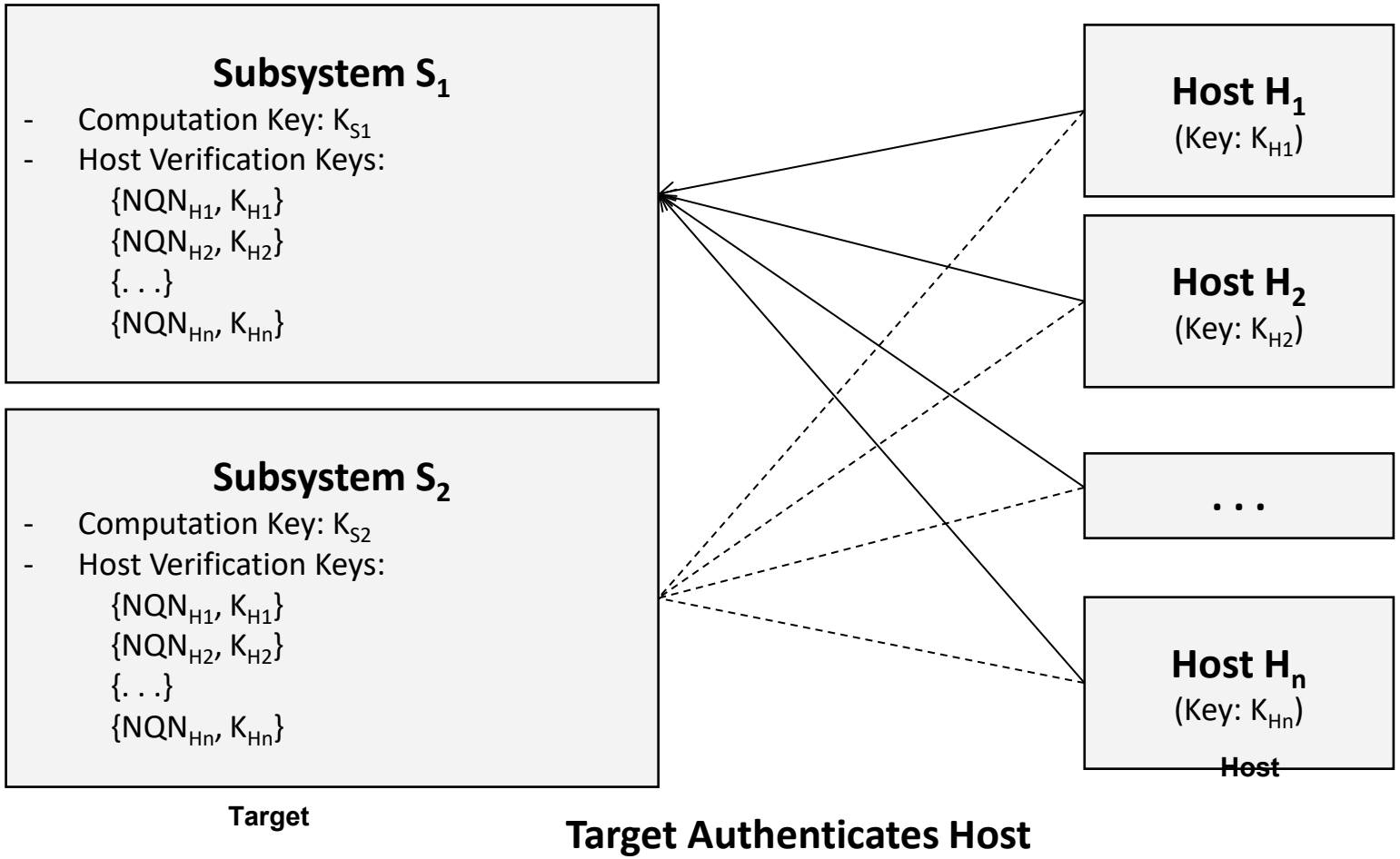


Security challenge

In-Band Authentication

Single Key per Entity– Multiple Hosts, Multiple Subsystems

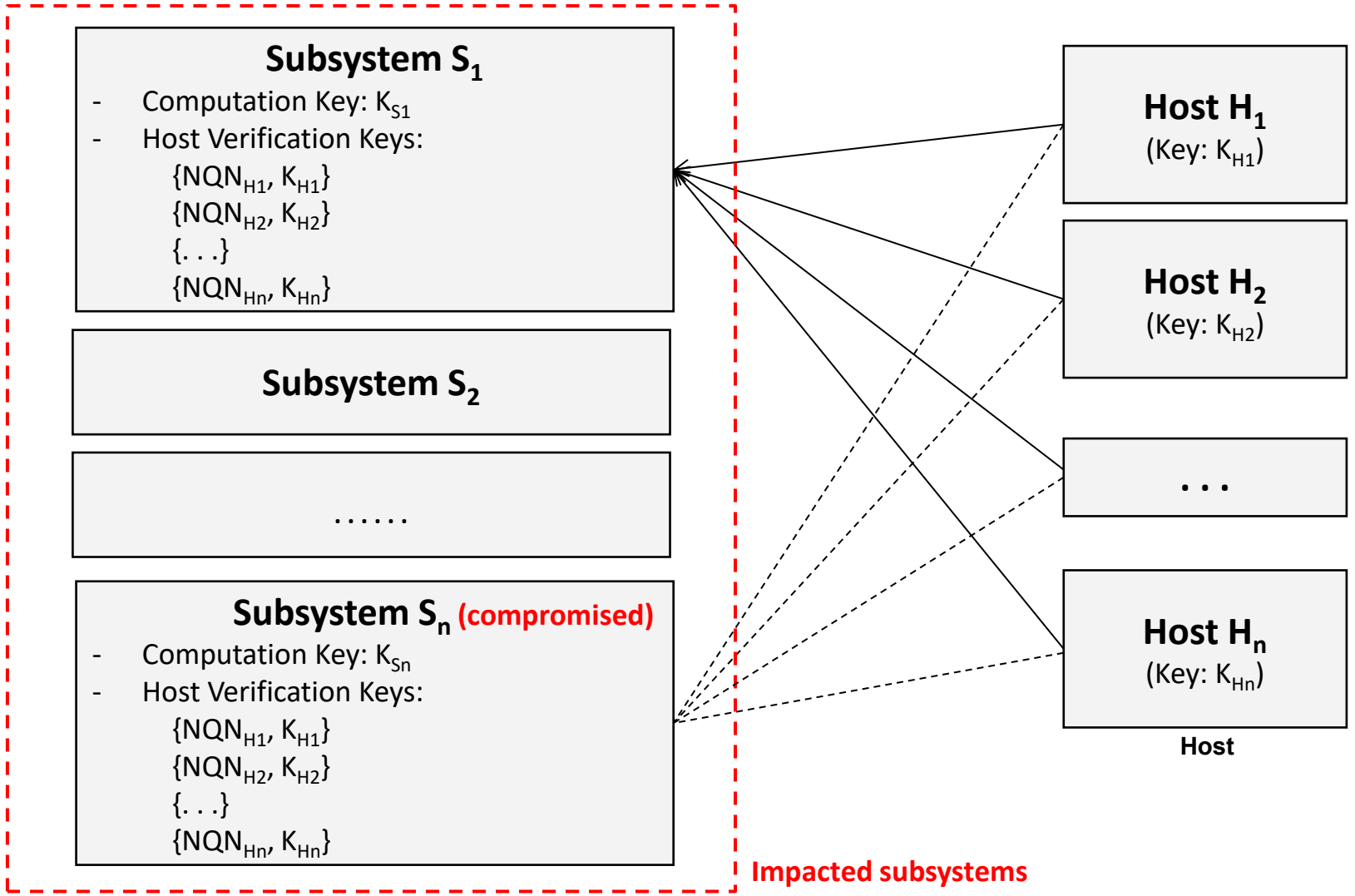
- Host uses same key for different subsytems – Example: For Host H1:K_{H1} for all subsytems
- Each entity (host/subsystem) has one unique key
- Subsystem authenticates Host (unidirectional authentication)
- Works in same way, when Host is authenticating Subsystems (bidirectional)



Target Authenticates Host

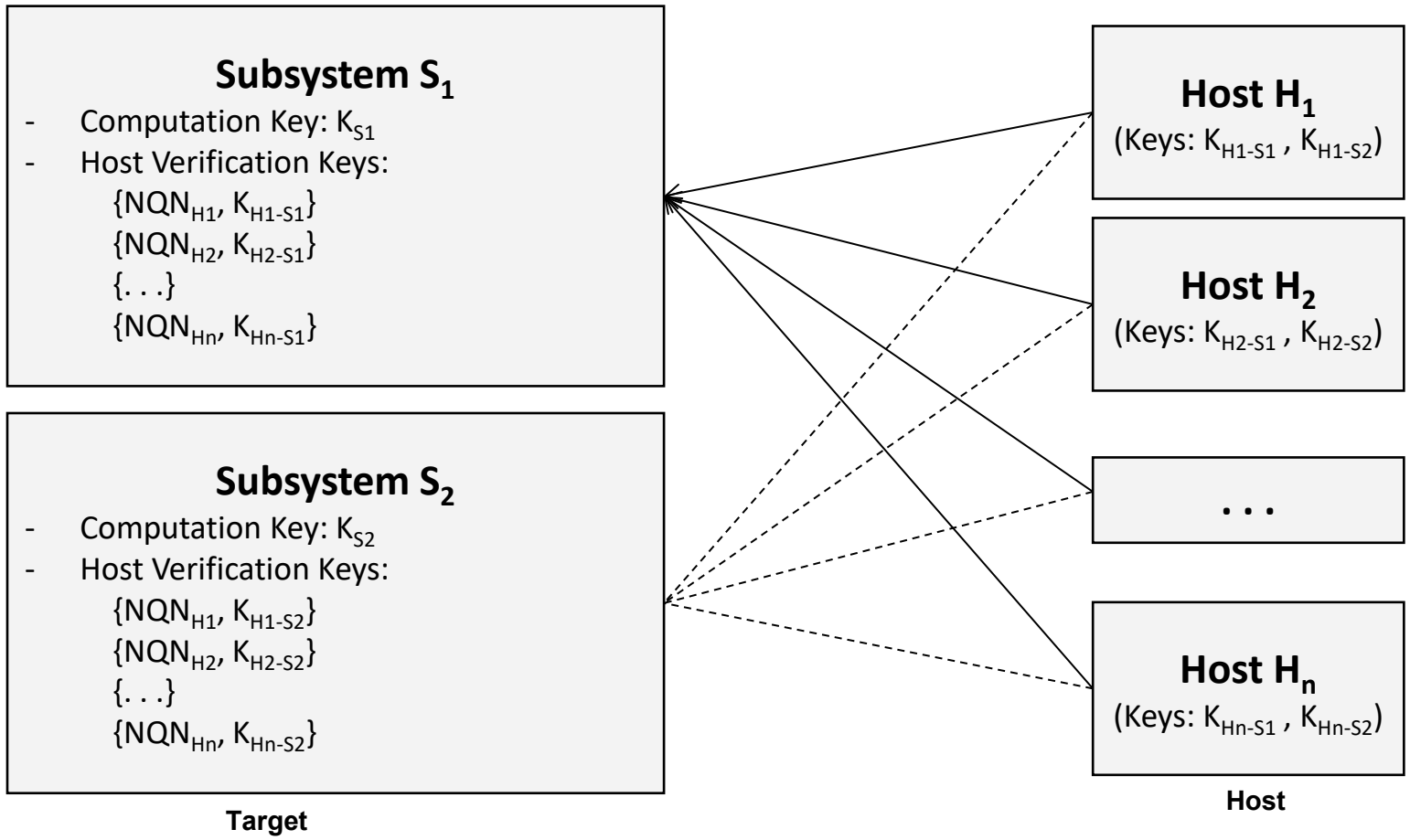
Single-Key Blast Radius – Entire Network compromised

- Attacker can impersonate any host to ALL the subsystems.
- S_n leaks K_{H1}.. K_{Hn} — hence attacker can impersonate any host to any subsystem.



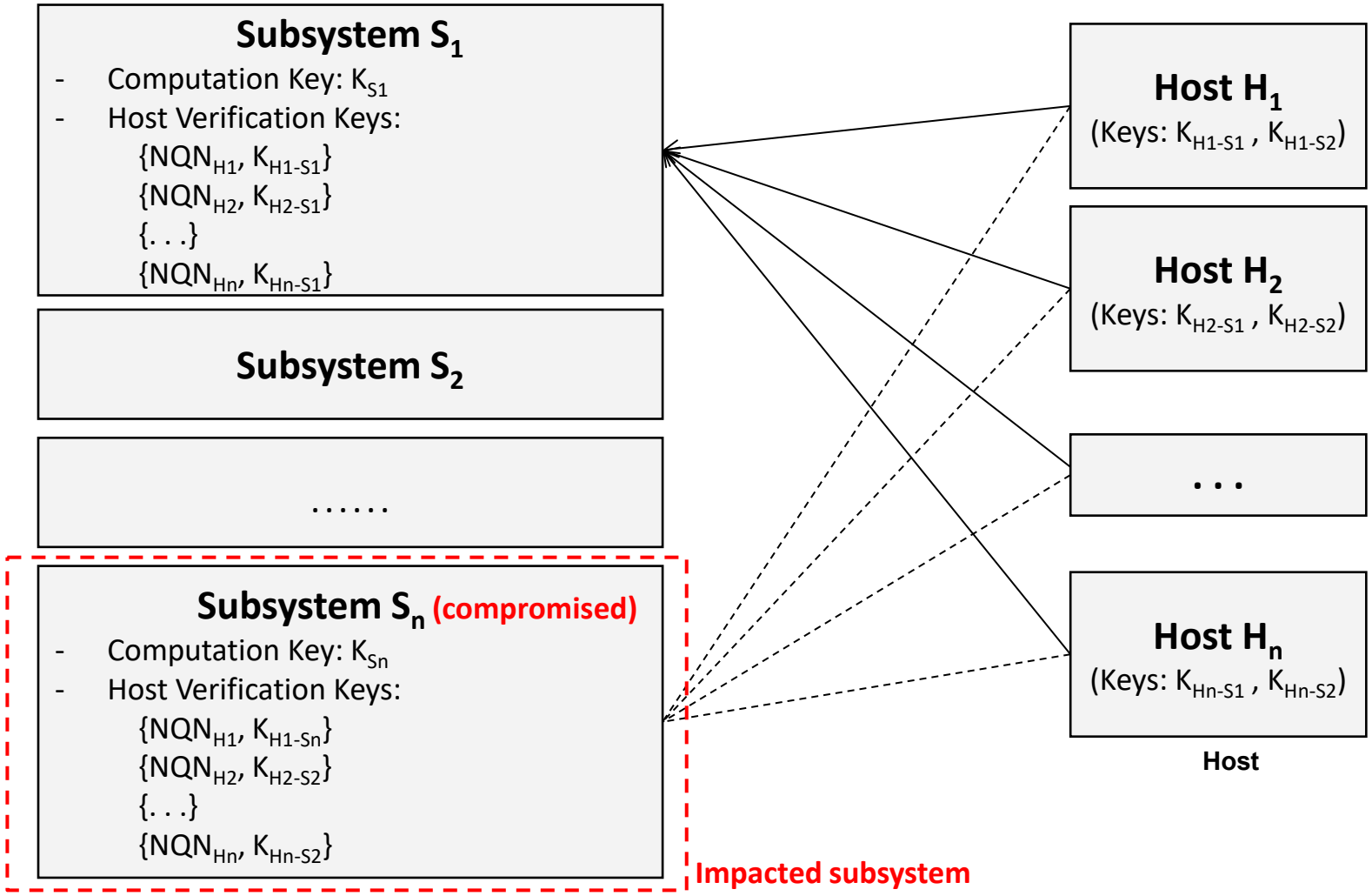
Multiple Keys – How it works

- Host uses different keys for different subsytems – Example: $K_{H1\text{-}S1}$ & $K_{H1\text{-}S2}$



Multiple Keys Blast Radius – Only Impacted Entity Compromised

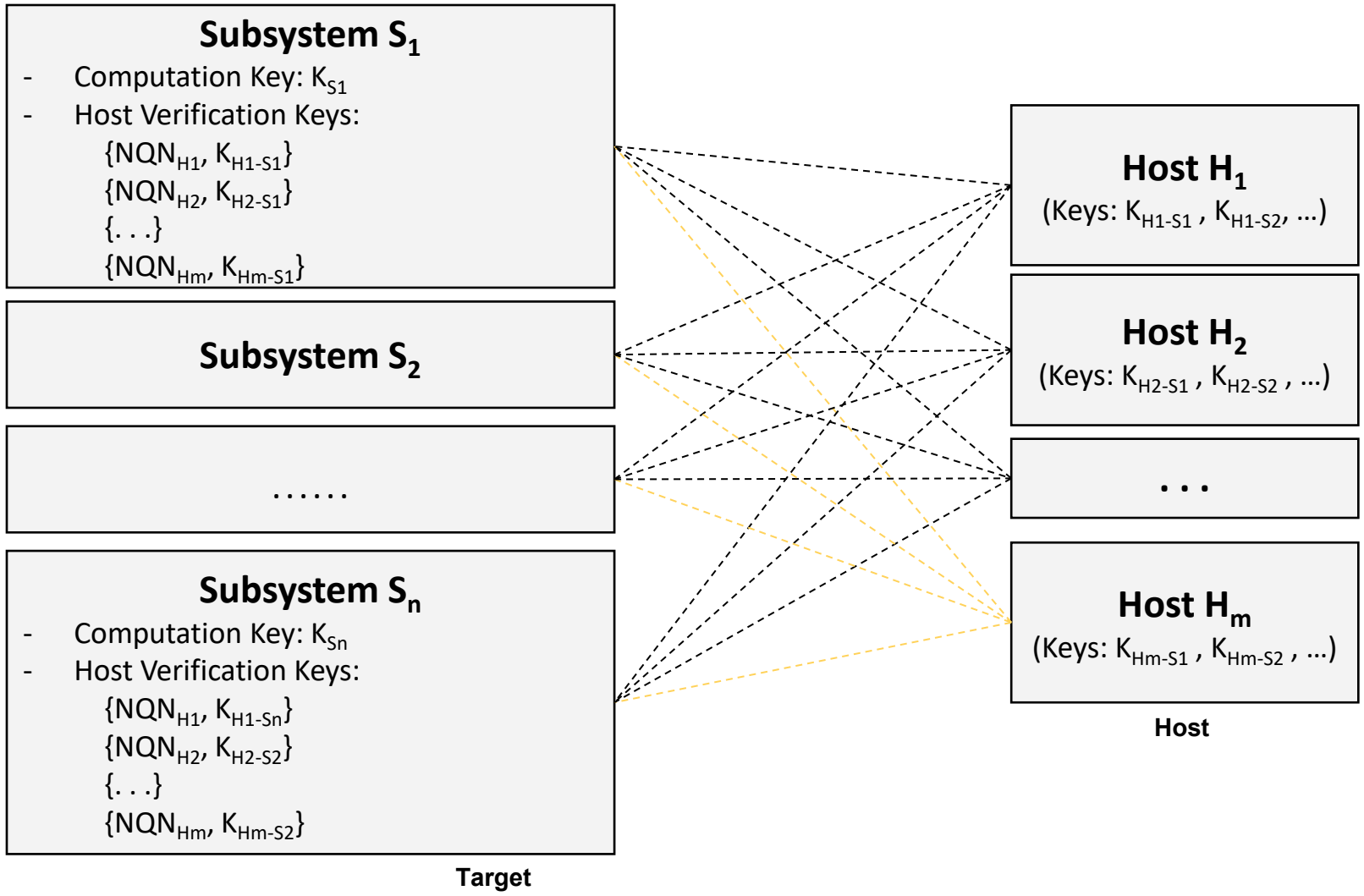
- Attacker can impersonate any host to only the compromised subsystem.
- S_n leaks $K_{H1\text{-}Sn}$.. $K_{Hn\text{-}Sn}$ — The attack does not affect other subsystems



Scaling challenge

Multiple Keys – Scaling problem

- For a network with 'N' Subsystems and 'M' Hosts, the number of keys to maintain are N x M
  - On addition of a new entity - Each entity needs to be provisioned with its own key



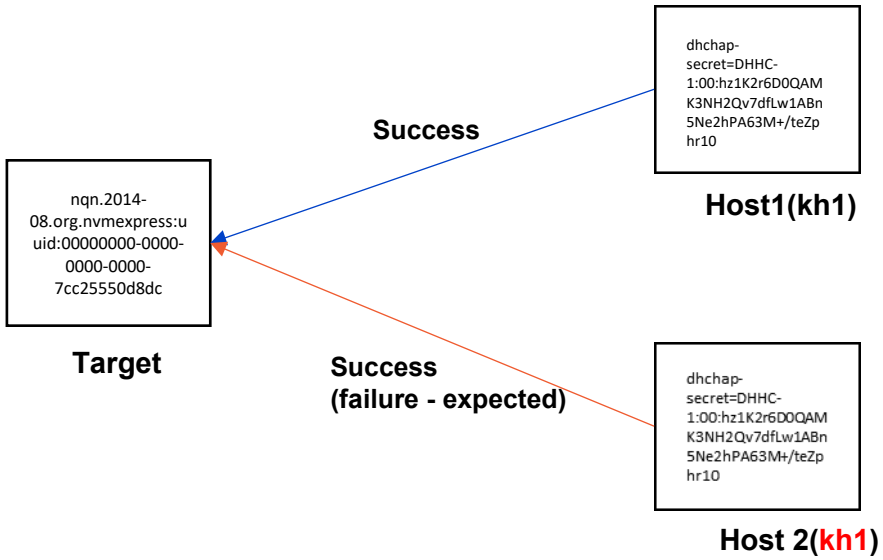
Our experiments and Observations

Generate DH-HMAC-CHAP keys with different Key length using nvme-cli

nvme gen-dhchap-key -n ${subsys_nqn} -m 1
DHHC-1:01:j1H4zDw/0KA2mBUNVPwYlIKUR6Xonstc7c7o1rayBl664Xyj:

nvme gen-dhchap-key -n ${subsys_nqn} -m 2
DHHC-1:02:2EMyv1YYc5nY0q00xV1Eq0m6Ikea5M2ZcXJGprqmG7Llvs8dZphBVEWuTsBJn3eTDE8O5g==:

nvme gen-dhchap-key -n ${subsys_nqn} -m 3
DHHC-1:03:mzed8yPrzUy7R79g2ZbLUmEhtGSdOdDsjMnpde3l2OVFkSq2F+oy72COkYsIVEWoevqrSEfWx3i0WurhlyQ+FMgMCTg=:



Security vulnerability of Single Key

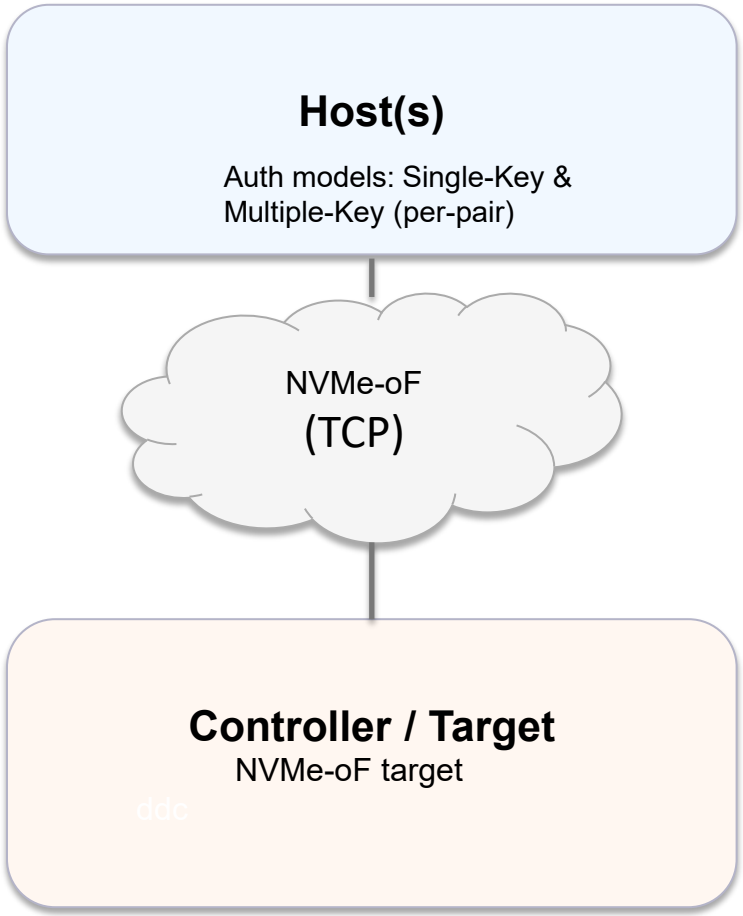
- **Connect host1 with DHCP Key(Host1)**

[root@Host1~]# nvme connect -t tcp -a 10.2.10.102 -s 4420 -n nqn.2023-01.nvmeof_test_target:subsystem1 --dhchap-secret=DHHC-1:00:hz1K2r6D0QAMK3NH2Qv7dfLw1ABn5Ne2hPA63M+/teZphr10:
connecting to device: nvme1

- **Connect host2 with host1 DHCP Key(Host2)**

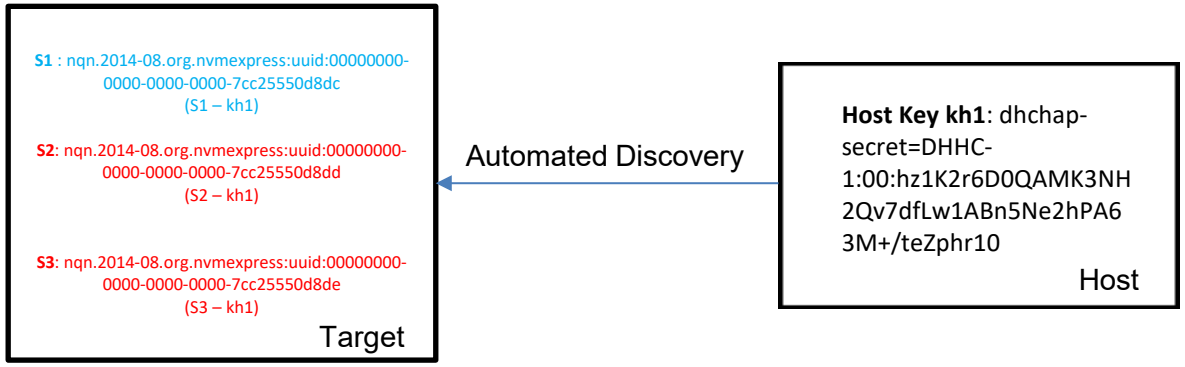
[root@Host2~]# nvme connect -t tcp -a 10.2.10.102 -s 4420 --hostnqn nqn.2014-08.org.nvmexpress:uuid:00000000-0000-0000-0000-7cc25550d8dc -n nqn.2023-01.nvmeof_test_target:subsystem1 --dhchap-secret=DHHC-1:00:hz1K2r6D0QAMK3NH2Qv7dfLw1ABn5Ne2hPA63M+/teZphr10:
connecting to device: nvme7

Our experiments and Observations



Single-Key model: worked consistently in our lab across both manual/automatic discovery methods

Multiple-Key (per host↔subsystem): observed authentication challenges with automatic discovery - When NVMe-oF target is configured with multiple keys -> automated discovery fails to discover more than one subsystem



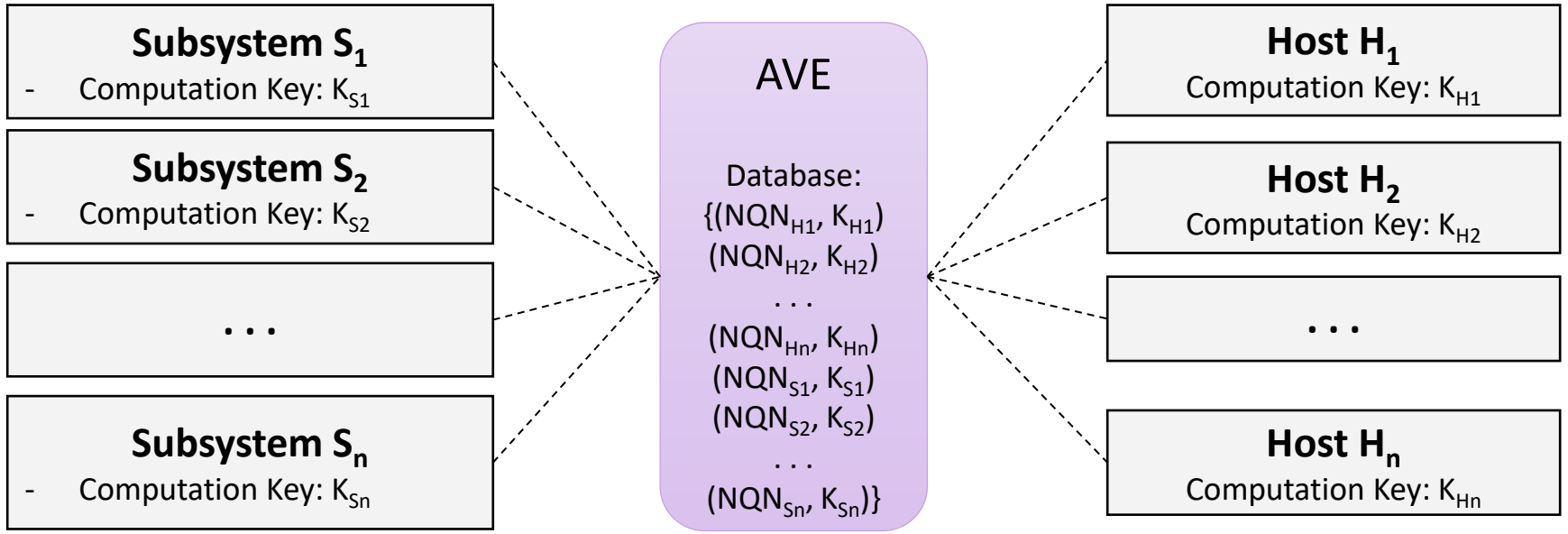
Provisioning at scale: Multiple Keys(per-pair Keys) increase operational overhead;

- Recently NVMe spec introduced AVE; AVE is a centralized Authentication verification entity
- We are considering an evaluation of AVE as a possible path to central verification and simpler operations

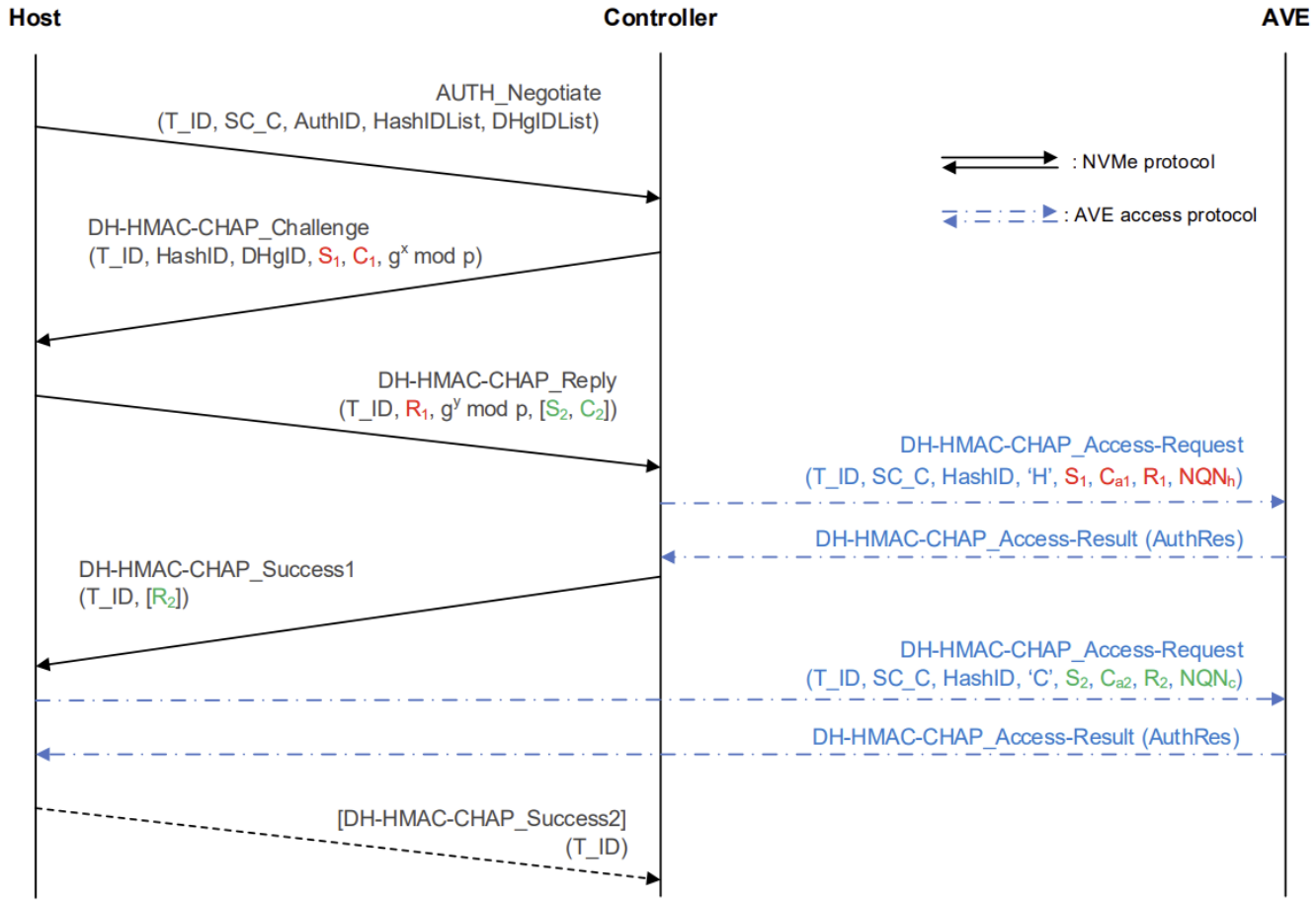
AVE Introduction in NVMe Spec

Authentication Verification Entity (AVE)

- An outsourcing of the verification process through a secure centralized entity having a complete database of keys
- Security problem: Each entity stores only its own key now. AVE in itself is also a secure entity(uses TLS).
- Scaling problem: AVE enables single key implementations on subsystems and hosts, greatly reducing the scaling problem.



DH-HMAC-CHAP with AVE

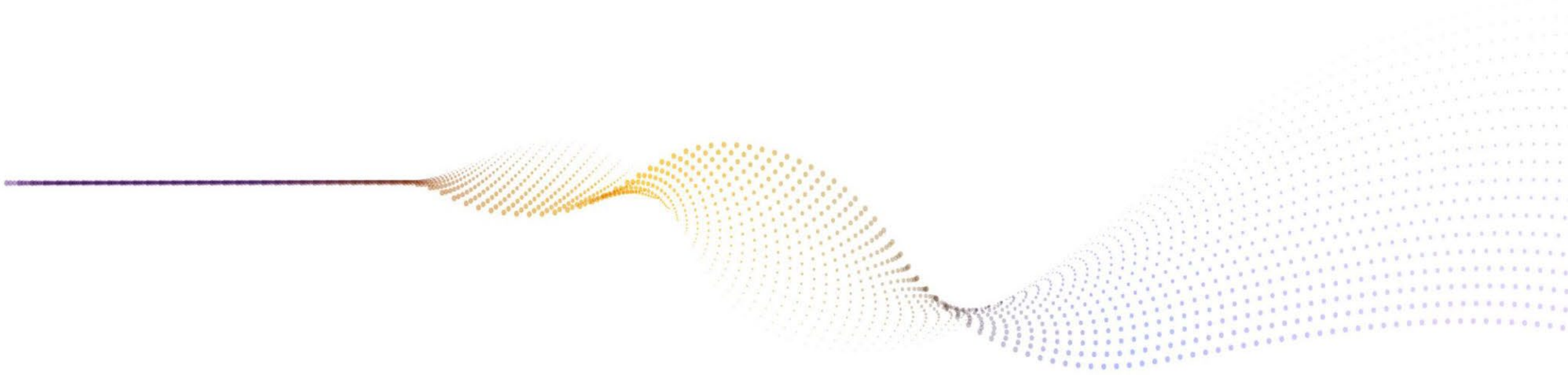


Our findings on NVMe-oF Authentication –

	Single Key	Multiple Key
Pre - AVE	- Not secure - Recommend: Migrate to AVE implementation	- Secure - Scaling/provisioning issue
AVE	- Secure - No issue with scaling	Not required

Conclusion

- **Key Takeaways**
  - Single-Key per entity ⇒ Security vulnerability; **Entire Network Compromised**
  - Multiple-Key per entity ⇒ Per-pair keys bound the blast radius but increase provisioning overhead
  - AVE centralizes verification: endpoints keep only their own keys; enables per-pair semantics at scale.



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

