V2X-HUB 보안 백서

1 개요

본 문서는 차량용 V2X 게이트웨이(**HUB**)와 차량 내부 하위 장치(**Equip** 노드) 전체를 클라우드 **Center**에 안전하게 결합하기 위한 보안 아키텍처를 제시한다. 주된 목적은 **완전히 동일하게 복제(clone)된 장치**가 차량 내부 네트워크에 물리적으로 삽입되는 최악의 내부 공격을 차단하는 것이다. 설계는 널리 사용되는 AES-256, SHA-3, HMAC 만으로 구현 가능하며, 복잡한 후양자(PQ) 알고리즘에 의존하지 않는다.

---

2 위협 모델 & 설계 목표

구분	공격자 능력
물리적 접근	Equip 장치를 탈-부착하고 완전 복제 장치를 추가할 수 있음
펌웨어 지식	정품 펌웨어·BindKey 를 모두 보유
네트워크 위치	차량 내부 통신을 모두 관찰·중계 가능

G1 — 복제·대체된 Equip 을 탐지·차단한다.\ **G2** — HUB↔Center 채널의 기밀성·무결성을 보장한다.\ **G3** — 장치 구성 정보(평문)를 HUB 외부에 노출하지 않는다.

---

3 장치 집합 토큰(DST : Device-Set Token)

```
DST = HUB_ID  ||  EQUIP_ID₀  || … ||  EQUIP_IDₙ
salt = 차량별 128-bit 무작위 값
```

DST 는 키 파생 시에만 메모리에 존재하며 즉시 삭제된다.

3.1 키 파생(단방향)

```
AuthKey = HKDF-SHA-256( DST || "AUTH" )        # HUB↔Center 인증 토큰
BindKey = AuthKey XOR SHA3-512( DST || salt )   # Equip 검증 키(가역식)
```

`AuthKey` 와 `DST` 로 언제든 `BindKey` 를 재계산할 수 있지만, `BindKey` 단독으로는 DST 를 유추할 수 없다.

3.2 참고 구현 예시

아래 파이썬-3 코드(라이브러리 **pyca/cryptography**)는 Equip 3개를 가진 샘플 HUB에 대해 위 파생 과정을 재현한다.

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from os import urandom

HUB_ID     = b"HUB_42"
EQUIP_IDS = [b"E01", b"E02", b"E03"]
salt       = urandom(16)
DST        = b"||".join([HUB_ID] + EQUIP_IDS)

hkdf = HKDF(algorithm=hashes.SHA256(), length=32, salt=None, info=b"AUTH")
AuthKey = hkdf.derive(DST)

digest = hashes.Hash(hashes.SHA3_512())
digest.update(DST + salt)
BindKey = bytes(a ^ b for a, b in zip(AuthKey, digest.finalize()[:32]))

print("AuthKey :", AuthKey.hex())
print("BindKey :", BindKey.hex())
```

리스트 1 – 샘플 토큰으로 생성된 32바이트 AuthKey·BindKey.\ 임베디드 C 환경에서는 mbedTLS/OpenSSL 의 HKDF·SHA3 함수를 사용하고 XOR 한 줄만 추가하면 동일 로직 구현이 가능하다.

---

4 단계별 워크플로

```
flowchart TD
  subgraph Phase 0 – 초기 스캔
    S0[HUB 장치 스캔 → DST 생성]
  end
  subgraph Phase 1 – 등록(온라인)
    S1[AuthKey·BindKey 파생]
    S2[HUB → Center : ID/PW + AuthKey]
    S3[Center, AuthKey 저장]
    S4[DST·AuthKey 삭제<br/>BindKey → Equip 전송]
  end
  subgraph Phase 2 – 로그인(시동마다)
    L1[HUB → Center : ID/PW]
    L2[Center, AuthKey 검증]<br/>L3[세션 키 합의]<br/>L4[AuthKey RAM 삭제]
  end
  subgraph Phase 3 – 운용
    O1{Center 연결 상태?}
    O2[온라인 ↔ MAC(Auth⊕Bind)]
    O3[오프라인 ↔ 읽기 전용·이벤트 버퍼링]
  end
  subgraph Phase 4 – 복제 탐지
    C1[브로드캐스트 폴]
    C2[ID 중복?]
    C3[주기 이상?]
```

```
    C4[구성 변화?]
    C5[알람 / 격리]
  end
  S0 --> S1 --> S2 --> S3 --> S4 --> L1
  L1 --> L2 --> L3 --> L4 --> O1
  O1 -->|Yes| O2 --> C1
  O1 -->|No| O3 --> C1
  C1 --> C2 -->|Yes| C5
  C1 --> C3 -->|Yes| C5
  C1 --> C4 -->|Yes| C5
```

그림 1 – 단계별 동작 흐름.

---

5 복제 탐지 로직

1. **ID Collision** — 동일 `EQUIP_ID` 응답이 2회 이상
2. **Timing Anomaly** — Equip 광고 주기 `P` 대비 `|Δ| > threshold`
3. **Composition Drift** — 신규/누락 ID → 새로운 DST′ 해시 불일치

---

6 암호 구성(AES / SHA-3)

프리미티브	용도
AES-256-GCM	링크 기밀성·무결성
SHA-3-512 / HKDF	단방향 KDF · BindKey 파생
HMAC-SHA-3-256	메시지 인증

---

7 보안 분석

- **잠복 복제** — 최초 브로드캐스트 미응답 ⟶ 이후 새 ID/중복 ID 검출
- **정품 제거 후 대체** — 하트비트 공백 + MAC 불일치
- **센터 오프라인** — AuthKey 부재 → 복제 장치 MAC 위조 불가
- **차량 간 키 재사용** — salt 구분으로 무효화

---

8 표준 비교

참조 표준	유사점	추가 가치
TCG DICE / Android DICE	해시 기반 디바이스 키	**다중 장치** 토큰 + 드리프트 알람
AUTOSAR SecOC	CAN/FlexRay MAC 보호	Center 고정 DST로 복제 장치 필터링
IEC 62351-11 SCADA	중앙 키 발급	자동차 내부 복제-대체 전용 탐지 논리 추가

9 가용성 대책

- **Grace Token (24h)** — 셀룰러 장애 시 HUB 기본 기능 유지
- **Dual Center** — 단일 장애점 제거
- 오프라인 이벤트는 재연결 시 일괄 검증

10 결론

본 아키텍처는 HUB-Equip 집합을 **Device-Set Token(DST)**으로 묶어 원본 구성 무결성을 보장하고, 완전 복제 장치를 실시간 탐지·차단한다. 구현은 AES-256과 SHA-3 계열만으로 가능해