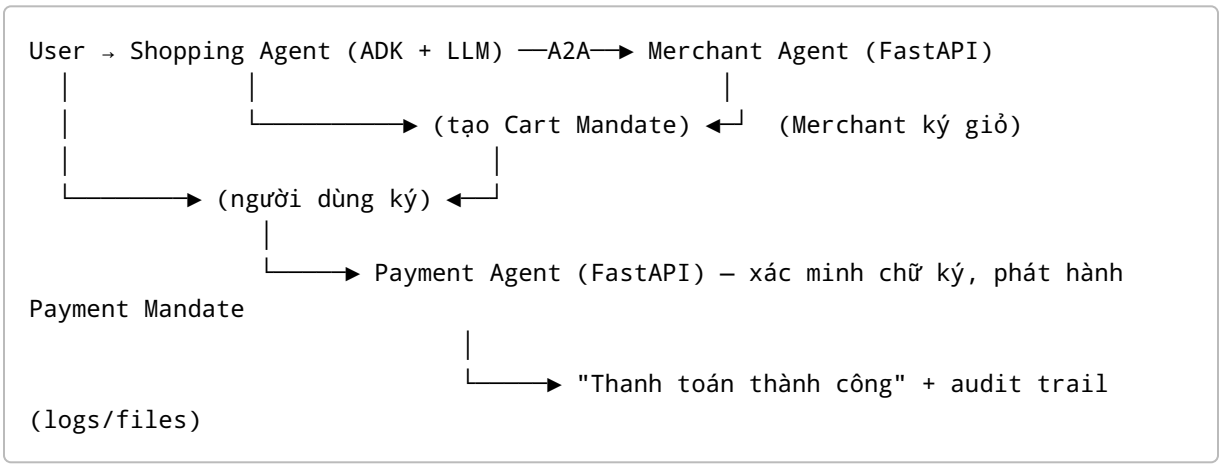
Bước 4 (AP2) — Thanh toán do Agent khởi xướng (on-prem)

Mục tiêu: hiện thực hoá luồng **thanh toán an toàn do agent khởi xướng** theo tinh thần AP2, hoàn toàn on-prem. Ta sẽ mô phỏng đầy đủ bằng **chứng cứ mật mã** (chữ ký số) và **chuỗi mandate**: - **Intent Mandate** (tuỳ chọn) – người dùng ủy quyền trước điều kiện mua. - **Cart Mandate** – giỏ hàng cuối cùng mà người dùng (và merchant) ký. - **Payment Mandate** – thông điệp thanh toán do Payment Agent phát hành để hoàn tất giao dịch.

> Bản này là **AP2-style**: giữ đúng tinh thần/định dạng dữ liệu cốt lõi, dùng **Ed25519** để ký/kiểm chứng, có audit trail, idempotency. Khi cần bạn có thể thay lớp ký bằng VC/JOSE/COSE.

---

1) Kiến trúc & luồng



Hai kiểu tác vụ: - **Human-present:** user duyệt giỏ → ký Cart Mandate → Payment Agent xác minh → thanh toán. - **Human-not-present:** user ký **Intent Mandate** trước (ví dụ điều kiện giá) → khi quote thoả → Shopping Agent tự động tạo Cart Mandate và thanh toán.

---

2) Cấu trúc dự án

```
adk-ap2-step4/
├─ shared/
│  ├─ __init__.py
│  ├─ ap2_schemas.py       # Pydantic models cho Mandate/Proof/Cart
│  └─ ap2_crypto.py        # Ed25519: sinh khoá, ký/verify, canonicalize JSON
├─ keys/
│  ├─ user_ed25519.pem     # private của người dùng (dev)
```

```
│  ├─ merchant_ed25519.pem # private của merchant (dev)
│  └─ payment_ed25519.pem  # private của payment agent (dev)
├─ payment_agent/
│  ├─ __init__.py
│  └─ server.py            # FastAPI: /ap2/intent, /ap2/checkout, /ap2/pay
├─ merchant_agent/
│  └─ server.py            # Bổ sung: ký cart (merchant_proof)
└─ shopping_agent/
   └─ agent.py             # Bổ sung: tạo & ký cart, gửi tới Payment Agent
```

---

3) Mô hình dữ liệu (shared/ap2_schemas.py)

```python
# shared/ap2_schemas.py
from __future__ import annotations
from pydantic import BaseModel, Field
from typing import List, Optional, Literal, Dict, Any
from datetime import datetime

class Proof(BaseModel):
    type: Literal["Ed25519Signature2020"] = "Ed25519Signature2020"
    key_id: str
    created: datetime
    signature: str  # base64/hex tuỳ chọn

class KeyDescriptor(BaseModel):
    key_id: str
    alg: Literal["Ed25519"] = "Ed25519"
    public_key: str  # base64/hex PEM-less

class LineItem(BaseModel):
    sku: str
    name: Optional[str] = None
    qty: int
    unit_price: float
    currency: str

class Cart(BaseModel):
    merchant_id: str
    buyer_id: str
    destination: str
    line_items: List[LineItem]
    shipping_fee: float
    total: float
    currency: str
```

```
    nonce: str
    created_at: datetime

class IntentMandate(BaseModel):
    mandate_type: Literal["intent"] = "intent"
    buyer_id: str
    conditions: Dict[str, Any]  # ví dụ: {"sku":"SKU002","max_price": 20.0}
    created_at: datetime
    proofs: List[Proof] = []

class CartMandate(BaseModel):
    mandate_type: Literal["cart"] = "cart"
    cart: Cart
    proofs: List[Proof] = []  # buyer_proof + merchant_proof (nếu có)

class PaymentMandate(BaseModel):
    mandate_type: Literal["payment"] = "payment"
    cart_hash: str  # hash của CartMandate
    processor_id: str
    created_at: datetime
    proofs: List[Proof] = []  # chữ ký Payment Agent
```

---

4) Ký/kiểm chữ ký (shared/ap2_crypto.py)

```
# shared/ap2_crypto.py
import json, base64, hashlib
from typing import Tuple
from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PrivateKey,
Ed25519PublicKey
from cryptography.hazmat.primitives import serialization

# —— JSON canonical (ổn định thứ tự) ——
def canonicalize(obj) -> bytes:
    return json.dumps(obj, sort_keys=True, separators=(",",
":")).encode("utf-8")

# —— PEM → key ——
def load_private_key(pem_path: str) -> Ed25519PrivateKey:
    with open(pem_path, "rb") as f:
        return serialization.load_pem_private_key(f.read(), password=None)

def load_public_key_from_private(pem_path: str) -> Ed25519PublicKey:
    return load_private_key(pem_path).public_key()
```

```
# ── Ký/verify ──
def sign_dict(priv_pem: str, payload: dict) -> str:
    sk = load_private_key(priv_pem)
    sig = sk.sign(canonicalize(payload))
    return base64.b64encode(sig).decode()

def verify_dict(pub: Ed25519PublicKey, payload: dict, b64sig: str) -> bool:
    try:
        sig = base64.b64decode(b64sig)
        pub.verify(sig, canonicalize(payload))
        return True
    except Exception:
        return False

# ── Hash cart để tham chiếu trong Payment Mandate ──
def hash_cart(cart: dict) -> str:
    return hashlib.sha256(canonicalize(cart)).hexdigest()
```

Dev tips: - Tạo khoá demo:

```
python - <<'PY'
from cryptography.hazmat.primitives.asymmetric.ed25519 import
Ed25519PrivateKey
from cryptography.hazmat.primitives import serialization
for who in ["user","merchant","payment"]:
    sk = Ed25519PrivateKey.generate()
    pem = sk.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption(),
    )
    open(f"keys/{who}_ed25519.pem","wb").write(pem)
PY
```

---

5) Merchant Agent — ký giỏ hàng (bổ sung vào merchant_agent/server.py)

```
# ... phần còn lại giữ nguyên từ Bước 3 ...
from shared.ap2_crypto import sign_dict
from shared.ap2_schemas import Cart, CartMandate
from datetime import datetime, timezone
```

```
MERCHANT_ID = "merchant_001"
MERCHANT_PRIV = "keys/merchant_ed25519.pem"

# Trong verb create_order (sau khi tính total/fee):
#   xây Cart + ký merchant_proof rồi trả về trong payload

# ví dụ, thay đoạn create_order trong Bước 3 bằng:
elif verb == "create_order":
    req = CreateOrderRequest(**payload)
    total = round(req.unit_price * req.qty + req.shipping_fee, 2)
    cart = Cart(
        merchant_id=MERCHANT_ID,
        buyer_id="buyer_001",  # demo, thực tế lấy từ session/user
        destination=req.destination,
        line_items=[LineItem(sku=req.sku, qty=req.qty,
unit_price=req.unit_price, currency=req.currency)],
        shipping_fee=req.shipping_fee,
        total=total,
        currency=req.currency,
        nonce=str(uuid.uuid4()),
        created_at=datetime.now(timezone.utc),
    )
    cm = CartMandate(cart=cart, proofs=[])
    # proof merchant
    base = cm.model_dump(exclude={"proofs"})
    sig = sign_dict(MERCHANT_PRIV, base)
    cm.proofs.append(Proof(key_id="merchant_k1",
created=datetime.now(timezone.utc), signature=sig))
    return _wrap(env, CreateOrderResponse(status="success",
order=None).model_dump() | {"cart_mandate": cm.model_dump()})
```

Gợi ý: nếu muốn giữ payload cũ, bạn cũng có thể trả song song `order` và `cart_mandate`.

---

6) Shopping Agent — tạo + ký Cart Mandate & gọi Payment Agent

```
# shopping_agent/agent.py (bổ sung)
import httpx, uuid
from datetime import datetime, timezone
from shared.ap2_crypto import sign_dict
from shared.ap2_schemas import CartMandate, Proof

USER_PRIV = "keys/user_ed25519.pem"
PAYMENT_URL = "http://localhost:8083"

# tool: finalize_checkout(cart_mandate_from_merchant)
```

```
def finalize_checkout(cart_mandate: dict) -> dict:
    # 1) user ký vào cart_mandate (thêm buyer_proof)
    cm = CartMandate(**cart_mandate)
    base = cm.model_dump(exclude={"proofs"})
    sig_user = sign_dict(USER_PRIV, base)
    cm.proofs.append(Proof(key_id="user_k1", created=datetime.now(timezone.utc),
signature=sig_user))

    # 2) gửi tới Payment Agent /ap2/pay để phát hành Payment Mandate
    with httpx.Client(timeout=15.0) as client:
        res = client.post(f"{PAYMENT_URL}/ap2/pay", json={"cart_mandate":
cm.model_dump()})
        res.raise_for_status()
        return res.json()
```

---

7) Payment Agent — xác minh & phát hành Payment Mandate (payment_agent/server.py)

```
# payment_agent/server.py
import uuid
from datetime import datetime, timezone
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from cryptography.hazmat.primitives import serialization
from shared.ap2_crypto import verify_dict, load_public_key_from_private,
hash_cart, sign_dict
from shared.ap2_schemas import CartMandate, PaymentMandate, Proof, KeyDescriptor

app = FastAPI(title="payment_agent")

# —— nạp public keys ——
USER_PRIV = "keys/user_ed25519.pem"
MERCHANT_PRIV = "keys/merchant_ed25519.pem"
PAYMENT_PRIV = "keys/payment_ed25519.pem"

USER_PUB = load_public_key_from_private(USER_PRIV)
MERCHANT_PUB = load_public_key_from_private(MERCHANT_PRIV)

class PayIn(BaseModel):
    cart_mandate: CartMandate

@app.post("/ap2/pay")
def pay(inmsg: PayIn):
```

```
    cm = inmsg.cart_mandate

    # 1) kiểm chứng có đủ 2 chữ ký: merchant & user
    base = CartMandate(**cm.model_dump()).model_dump(exclude={"proofs"})
    has_user = has_merchant = False
    for p in cm.proofs:
        if p.key_id.startswith("user"):
            if verify_dict(USER_PUB, base, p.signature):
                has_user = True
        if p.key_id.startswith("merchant"):
            if verify_dict(MERCHANT_PUB, base, p.signature):
                has_merchant = True
    if not (has_user and has_merchant):
        raise HTTPException(status_code=400, detail="invalid proofs")

    # 2) phát hành Payment Mandate (ký bởi Payment Agent)
    cart_hash = hash_cart(base)
    pm = PaymentMandate(
        cart_hash=cart_hash,
        processor_id="payment_agent_001",
        created_at=datetime.now(timezone.utc),
        proofs=[],
    )
    pm_base = pm.model_dump(exclude={"proofs"})
    sig = sign_dict(PAYMENT_PRIV, pm_base)
    pm.proofs.append(Proof(key_id="payment_k1",
created=datetime.now(timezone.utc), signature=sig))

    # 3) (demo) coi như thanh toán thành công → trả về receipt
    return {"status": "success", "payment_mandate": pm.model_dump(),
"receipt_id": str(uuid.uuid4())}

# uvicorn payment_agent.server:app --host 0.0.0.0 --port 8083
```

---

8) Chạy & kiểm thử

1) **Chạy Payment Agent**

```
uvicorn payment_agent.server:app --host 0.0.0.0 --port 8083
```

2) **Chạy Merchant Agent** (đã bổ sung ký cart)

```
uvicorn merchant_agent.server:app --host 0.0.0.0 --port 8082
```

3) **Chạy Shopping Agent** (ADK `shopping_agent` có tool `finalize_checkout`)

```
adk web
# Chat: "Báo giá SKU002 x2 ship Hanoi" → quote → reserve → create_order (nhận
cart_mandate)
# Chat: "Thanh toán giỏ này" → gọi tool finalize_checkout → nhận payment_mandate
+ receipt
```

4) **Test trực tiếp Payment Agent** (bỏ qua LLM) - Lấy `cart_mandate` (đã có merchant_proof + user_proof) rồi POST vào `/ap2/pay`.

---

9) Definition of Done (PASS Bước 4)

- [ ] **Chuỗi mandate đầy đủ**: (tuỳ chọn) Intent Mandate → Cart Mandate (user+merchant proof) → Payment Mandate.
- [ ] **Xác minh chữ ký**: Payment Agent **bắt buộc** thấy đủ 2 proof hợp lệ trước khi phát hành Payment Mandate.
- [ ] **Idempotency**: sử dụng `cart.nonce` + `cart_hash` để chống trả tiền lặp (ghi log/đánh dấu processed hashes).
- [ ] **Audit trail**: log JSON (hoặc ghi file) cho mỗi bước; verify lại chữ ký được từ log.
- [ ] **On-prem 100%**: mọi thành phần chạy cục bộ; model LLM có thể là vLLM/Ollama.
- [ ] **E2E happy path**: quote → reserve → create_order → pay chạy trơn tru nhiều lần; lỗi nhập liệu trả thông báo rõ ràng.

---

10) Bảo mật & mở rộng

- **Khoá & niêm cất**: trong prod, private key nên để HSM/KMS; giao tiếp nội bộ dùng **mTLS/JWT/NetworkPolicy**.
- **Mô hình VC/JOSE/COSE**: thay `Proof` bằng VC (W3C) hoặc JWS/COSE Signed Object, thêm `kid`/`alg` chuẩn.
- **Payment rails**: nối ra sandbox thật (card/crypto) ở lớp Payment Agent, giữ nguyên validate proof trước khi gọi gateway.
- **Dispute & replay**: log `cart_hash`, `payment_mandate` và thời điểm; từ đó có thể đối chiếu tranh chấp.
- **Quan sát**: Prometheus metrics `ap2_requests_total`, `ap2_verify_fail_total`, `ap2_latency_seconds`.

---

11) K8s gợi ý triển khai

- Mỗi agent 1 **Deployment**; **Service** nội bộ; **NetworkPolicy** chỉ cho phép Shopping↔Merchant↔Payment.

- Mount `keys/` qua **Kubernetes Secret** (PEM base64) vào đúng container path, quyền `0400`.
- ReadinessProbe: `/healthz`; LivenessProbe: `/livez` cho 2 FastAPI.
- Ingress (nếu cần) chỉ expose Shopping (UI/API); Merchant/Payment nội bộ cluster.

---

12) Bước tiếp theo

- Chuẩn hoá `Cart` thành **line_items[]** đầy đủ (tax/discount), map 1-1 sang bản AP2 chính thức.
- Bổ sung **Human-not-present**: xây `IntentMandate` (buyer ký trước) + job watcher tự động chốt đơn khi quote phù hợp.
- Viết **pytest e2e**: tạo cart → merchant ký → user ký → POST /ap2/pay → xác minh trả về `payment_mandate` hợp lệ.