TrustedPAI

Documentazione Tecnica dei Protocolli

Enrico De Vito - enxdev / 0xENX

Introduzione: perché questo documento esiste, e come leggerlo

Questo documento spiega cos'è TrustedPAI, perché esiste, e come funziona ogni singolo pezzo (protocolli, motore di rischio, funzionalità, sicurezza, architettura), con diagrammi, esempi pratici passo-passo, domande frequenti, e lo stato reale del progetto.

Questo documento nasce da una richiesta semplice sulla carta e in realtà molto ambiziosa: spiegare TrustedPAI a chi parte da zero, ma spiegarlo davvero fino in fondo. Non un riassunto per addetti ai lavori, non una brochure commerciale, ma qualcosa che si possa leggere dall'inizio alla fine e, arrivati all'ultima pagina, aver capito ogni pezzo: cosa fa il progetto, perché esiste, come funziona internamente ognuno dei protocolli che tratta, cosa succede davvero quando qualcosa va storto, e, altrettanto importante, cosa oggi ancora non fa e perché.

Non è un documento tecnico nel senso stretto del termine (non è pensato per chi deve scrivere codice contro l'API), e non è nemmeno un documento puramente divulgativo che resta in superficie. È qualcosa nel mezzo: ogni concetto tecnico viene introdotto con un'analogia semplice prima, spesso accompagnata da un diagramma, e solo dopo spiegato nel dettaglio reale, con i nomi veri degli standard, i nomi veri dei campi, i numeri veri delle soglie, così chi legge non deve scegliere tra “capire in generale” e “capire davvero”.

La struttura segue un percorso preciso, pensato per essere letto in ordine la prima volta, e poi consultato a pezzi le volte successive. Si parte dal “cos'è” in una frase, si allarga al problema che il progetto risolve, si fa un passo indietro a capire da dove arriva tutto questo (il contesto storico), poi si entra nel meccanismo vero e proprio (passo per passo, protocollo per protocollo, con esempi concreti dall'inizio alla fine) per poi passare alle funzionalità intorno al nucleo, alla sicurezza, alle domande che chiunque si farebbe leggendo tutto questo per la prima volta, e infine allo stato onesto del progetto: cosa c'è oggi, cosa manca, e perché manca deliberatamente.

I quattro punti più difficili da capire solo a parole (il flusso completo di una transazione, la struttura dei tre mandati di AP2, la logica con cui si arriva a un verdetto, e come sono collegati i pezzi tecnici tra loro) sono accompagnati da un diagramma, oltre che dal testo: se una frase risulta poco chiara al primo passaggio, spesso l'immagine vicina la rende immediata.

Un'ultima nota prima di iniziare: il commercio agentico, cioè i pagamenti fatti da agenti AI per conto di persone, è un campo nato letteralmente nel 2025 e ancora in piena evoluzione mentre queste righe vengono scritte, a metà del 2026. Alcuni dettagli qui dentro (versioni beta di protocolli, numeri di partner, stato di adozione) cambieranno nei prossimi mesi. Quello che invece resta stabile, ed è il vero cuore di questo documento, è il PERCHÉ tutto questo esiste e COME ragiona un sistema come TrustedPAI di fronte a un pagamento fatto da una macchina: quella logica di fondo cambia molto più lentamente dei singoli dettagli tecnici.

In fondo al documento trovi anche un glossario completo in ordine alfabetico e una sezione di domande frequenti: se durante la lettura un termine non è chiaro, o se una domanda ti viene in mente prima ancora di arrivarci nel testo, quelle due sezioni sono pensate apposta per essere consultate anche fuori ordine.

1. Cos'è TrustedPAI: in una frase, e poi per esteso

TrustedPAI è un servizio che controlla, PRIMA che un pagamento venga eseguito, se un “agente” di intelligenza artificiale che sta comprando qualcosa per conto di una persona ha davvero il permesso di farlo, e se quella transazione sembra rischiosa o sospetta.

In pratica è un guardiano: non muove mai i soldi, ma dà un giudizio (via libera, da controllare a mano, oppure blocca) prima che il pagamento vada a buon fine. Il paragone più calzante è quello del buttafuori all'ingresso di un locale: controlla i documenti e decide chi entra, ma non tocca mai la cassa.

Detto in modo un po' più tecnico, ma ancora accessibile: TrustedPAI è un'API, cioè un servizio a cui altri programmi si collegano via internet per scambiarsi informazioni senza intervento umano, di risk-screening (valutazione del rischio) specializzata nel commercio agentico, cioè in quella fetta di pagamenti online dove chi “clicca compra” non è più necessariamente un umano davanti a uno schermo, ma un programma AI che agisce per suo conto.

Una domanda che sorge spontanea a questo punto: perché serve un servizio dedicato, e non basta il sistema antifrode che un negozio online usa già oggi per i pagamenti umani normali? La risposta breve, che verrà approfondita nella sezione 2, è che i sistemi antifrode tradizionali sono costruiti attorno a segnali comportamentali umani: velocità di battitura, movimento del mouse, tempo passato su una pagina, pattern di navigazione. Sono segnali che un agente AI semplicemente non genera, o genera in modo completamente diverso. Un sistema pensato per riconoscere un umano sospetto non è automaticamente bravo a riconoscere un agente AI sospetto: sono due problemi imparentati, ma diversi.

2. Il problema che risolve: cos'è il “commercio agentico”

Fino a poco tempo fa, ogni pagamento online partiva da un gesto umano: una persona vera che clicca “compra” su un sito, dopo aver guardato il prezzo e il prodotto. Quel click era, di fatto, la prova che qualcuno aveva davvero deciso di comprare, in quel momento, a quel prezzo.

Oggi sempre più spesso non è più così. Un assistente AI (ChatGPT, Gemini, o un agente costruito su misura da un'azienda) può comprare qualcosa al posto di una persona, magari mentre quella persona non sta nemmeno guardando lo schermo. Due esempi concreti, molto diversi tra loro:

> *Scenario “presente”: chiedi a un assistente “trovami delle scarpe da corsa bianche sotto i 150€”, l'assistente ti mostra un carrello con due paia, tu lo guardi e confermi lì per lì. L'umano è presente e approva nel momento esatto dell'acquisto.*
>
> *Scenario “delegato”: dici una volta sola “comprami un biglietto per il concerto appena escono in vendita, non più di 100€, solo se è nel weekend”, e l'agente esegue l'acquisto vero settimane dopo, da solo, senza che nessuno stia guardando in quel momento preciso.*

Il secondo scenario è quello nuovo, e crea un problema che prima non esisteva: come fa il negozio a sapere che quell'agente ha DAVVERO il permesso della persona vera, per QUELL'acquisto specifico? Come si accorge se l'agente ha sbagliato, se qualcuno ha manomesso l'ordine lungo il tragitto, o se è un abuso deliberato (un agente compromesso, o un'istruzione ambigua interpretata male)? Prima, il click umano era di per sé una garanzia minima contro tutto questo. Ora quel controllo può mancare del tutto: serve quindi qualcosa d'altro al suo posto. È esattamente lo spazio in cui opera TrustedPAI, e per cui sono nati i cinque protocolli descritti nella sezione 5.

Vale la pena essere concreti su cosa può andare storto, perché sono categorie di rischio molto diverse tra loro e vengono confuse spesso:

- **Errore in buona fede dell'agente**: l'AI interpreta male un'istruzione ambigua (“compra qualcosa di carino per il compleanno di mia figlia” lascia troppo spazio) e acquista qualcosa che la persona non avrebbe scelto. Non è un attacco, è un fraintendimento.

- **Agente compromesso**: qualcuno riesce a manipolare l'agente dall'esterno (per esempio con contenuti ostili nascosti in una pagina web che l'agente legge) per fargli eseguire un acquisto che l'utente non ha mai chiesto.
- **Manomissione lungo il tragitto**: l'ordine parte corretto ma qualcuno, tra l'agente e il negozio, altera il prezzo o il destinatario prima che il pagamento venga eseguito.
- **Furto o riutilizzo di un'autorizzazione**: un'autorizzazione di pagamento legittima, già usata una volta, viene intercettata e riproposta per pagare una seconda volta (il “replay attack” del glossario).
- **Abuso deliberato**: qualcuno costruisce apposta un agente per sfruttare falle nei negozi (per esempio comprando in massa un prodotto in offerta limitata, o generando ordini fasulli per drenare crediti promozionali).

Ognuna di queste categorie richiede un controllo leggermente diverso, ed è per questo che TrustedPAI non fa UNA sola verifica, ma una combinazione: prima la crittografia (per essere sicuri che l'autorizzazione sia autentica e non manomessa, copre le categorie 2, 3 e 4), poi il motore di rischio basato su regole e storico (per intercettare pattern sospetti anche quando la firma è perfettamente valida, copre le categorie 1 e 5).

3. Un po' di storia: come siamo arrivati fin qui

Per capire perché nel 2025-2026 sono nati quasi contemporaneamente cinque protocolli diversi per far pagare gli agenti AI, aiuta guardare rapidamente da dove viene tutto questo: non è un fulmine a ciel sereno, è il punto di arrivo (provvisorio) di una serie di passaggi che si sono susseguiti nell'arco di circa trent'anni.

Dai numeri di carta digitati a mano al “clic per comprare”

Negli anni '90 e 2000, comprare online voleva dire digitare a mano, ogni volta, il numero della propria carta di credito su un modulo web. Era lento, era rischioso (il numero della carta viaggiava in giro, in chiaro o quasi, tra siti diversi), e ogni sito doveva costruirsi da solo il proprio sistema di pagamento. Pian piano sono arrivati intermediari come PayPal e poi Stripe, che hanno reso il pagamento “un servizio” che un sito poteva integrare senza dover gestire direttamente i dati sensibili delle carte; questo è già un primo parallelo diretto con quello che fa oggi TrustedPAI per il rischio: prendere una responsabilità delicata e specializzata, e offrirla come servizio a chiunque ne abbia bisogno, invece di farla reinventare da ogni singola azienda.

La lotta contro la frode, e la nascita della “presenza umana” come prova

Con la crescita dell'e-commerce è cresciuta in parallelo la frode con carte rubate. La risposta del settore è stata costruire livelli di verifica dell'identità sempre più sofisticati: il codice di sicurezza sul retro della carta (CVV), la verifica dell'indirizzo di fatturazione (AVS), e poi il cosiddetto “3-D Secure” (quello per cui, pagando online, a volte arriva un codice via SMS o una notifica sull'app della banca da confermare). Tutti questi sistemi hanno in comune una cosa: presumono che dietro allo schermo ci sia una persona fisica, in quel momento, capace di guardare un messaggio e rispondere. È esattamente questa presunzione a non reggere più quando chi “clicca compra” è un programma.

L'economia delle API, e i pagamenti che diventano programmabili

Nel decennio successivo, aziende come Stripe hanno reso i pagamenti “programmabili”: non più solo un modulo da compilare, ma un insieme di funzioni che un programmatore può richiamare direttamente dal proprio codice. Questo ha aperto la strada a tutto ciò che oggi diamo per scontato: abbonamenti automatici, marketplace con pagamenti divisi tra più parti, checkout integrati in app di terzi. Ha anche gettato, senza che fosse lo scopo dichiarato, le basi tecniche su cui poggia oggi il commercio agentico: se un pagamento può essere

richiamato da codice scritto da un umano, prima o poi qualcuno avrebbe fatto in modo che venisse richiamato da codice scritto (o eseguito) da un'intelligenza artificiale.

L'arrivo degli agenti AI capaci di agire, non solo di rispondere

Tra il 2023 e il 2025 i modelli linguistici di grandi dimensioni (LLM, come GPT o Gemini) sono passati dal semplice “rispondere a una domanda” al poter usare strumenti esterni, navigare siti web, ed eseguire sequenze di azioni autonome per raggiungere un obiettivo assegnato da un umano, quella che nel settore si chiama, appunto, “AI agentica”. Nel momento in cui questi agenti hanno iniziato a poter compilare moduli, cliccare pulsanti e completare processi web autonomamente, comprare qualcosa online è diventata una delle prime applicazioni pratiche ed economicamente rilevanti. Con essa è arrivato, quasi subito, il problema descritto nella sezione 2: come si fa a sapere che un pagamento fatto da un agente è davvero autorizzato?

2025-2026: la corsa ai protocolli

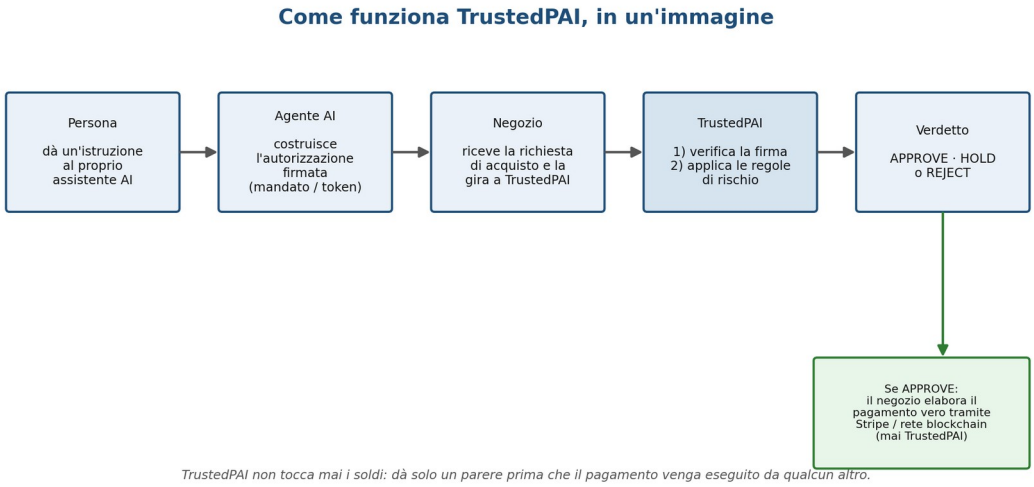
Nell'arco di poco più di un anno, quasi in parallelo, diversi grandi attori del settore hanno pubblicato i propri protocolli per rispondere esattamente a questa domanda, ognuno partendo dai propri punti di forza esistenti: Google (con AP2) ha portato l'esperienza in identità digitale e credenziali verificabili; Stripe e OpenAI (con ACP) hanno portato l'infrastruttura di pagamento già usata da milioni di negozi e l'esperienza diretta con “Buy it in ChatGPT”; Coinbase (con x402) ha portato l'esperienza in stablecoin e blockchain per pagamenti istantanei e a bassissimo costo; Stripe e Tempo (con MPP, arrivato più tardi, a marzo 2026) hanno affrontato il caso specifico dei micropagamenti a raffica. È un momento simile, per certi versi, ai primi anni di internet quando convivevano più protocolli concorrenti prima che uno o pochi si affermassero come standard di fatto, con la differenza che stavolta le grandi aziende coinvolte sono partite subito collaborando su alcuni pezzi (per esempio la compatibilità dichiarata tra UCP e AP2) invece di ignorarsi a vicenda.

È in questo momento storico preciso (un settore giovanissimo, con più standard che si stanno ancora consolidando, dove nessuno ha ancora “vinto” definitivamente) che TrustedPAI ha scelto la propria posizione: non scommettere su un solo protocollo, ma essere neutrale e supportarli tutti, così da restare utile indipendentemente da quale standard finirà per prevalere, o se continueranno a convivere più standard specializzati per casi d'uso diversi (cosa

più probabile, vista la specializzazione di ciascuno: AP2 per mandati complessi, x402 per micropagamenti istantanei, MPP per sessioni continue).

4. Come funziona TrustedPAI in pratica, passo per passo

Seguiamo un esempio concreto, dall'inizio alla fine, con lo scenario “agente delegato” di prima. Prima il colpo d'occhio, poi il dettaglio passo per passo:



Il percorso completo: dalla richiesta della persona al verdetto di TrustedPAI, fino al pagamento vero.

1. L'agente AI, arrivato il momento di comprare il biglietto del concerto, si presenta al negozio (il “merchant”) con la propria autorizzazione, un “mandato” (in parole povere: un permesso scritto, con una firma digitale impossibile da falsificare, che dimostra che l'utente ha davvero autorizzato quell'acquisto), in uno dei formati descritti nella sezione 5 (AP2, ACP, x402 o MPP).
2. Il negozio, invece di processare subito il pagamento, manda i dettagli della transazione a TrustedPAI: una singola chiamata API, contenente il mandato/l'autorizzazione, l'importo, e l'identificativo dell'agente.
3. TrustedPAI fa due cose, sempre in questo ordine: prima verifica la firma crittografica dell'autorizzazione (per essere sicuro che non sia falsa, non sia stata alterata, e non sia già stata usata prima); solo se questo controllo passa, applica poi le regole di rischio configurate dal negozio (vedi sezione 8).
4. TrustedPAI risponde con un verdetto, in genere in pochi millisecondi: APPROVE (via libera), HOLD (sospetto, va controllato da una persona prima di procedere), oppure REJECT (blocca, non procedere), insieme a un punteggio di rischio da 0 a 100 e una spiegazione testuale del perché.

5. Solo a quel punto il negozio decide se finalizzare il pagamento vero e proprio, che avviene sempre tramite un vero processore di pagamenti (Stripe, una rete blockchain, ecc.), mai tramite TrustedPAI stesso.

Punto chiave da tenere a mente per tutto il resto del documento: TrustedPAI non tocca mai i soldi, in nessuna fase di questo processo.

Chi paga per usare TrustedPAI? Il negozio, oppure il PSP che gestisce i pagamenti per conto del negozio (come Stripe). Non paga chi possiede l'agente AI (OpenAI, Google…), e non paga l'infrastruttura di pagamento stessa (Stripe, le reti blockchain): il cliente di TrustedPAI è sempre chi RICEVE il pagamento, non chi lo genera.

Un dettaglio di filosofia importante, che riguarda cosa succede quando qualcosa va storto DENTRO TrustedPAI (non nella transazione, proprio nel servizio stesso, es. il database è momentaneamente irraggiungibile): il sistema non risponde mai con un errore secco. Risponde comunque con 200 (successo tecnico) e un verdetto HOLD esplicito, con la motivazione “valutazione automatica fallita, trattenuto per sicurezza, richiede revisione manuale”. L'idea è che il negozio abbia sempre qualcosa su cui agire (fermarsi e controllare a mano), invece di restare senza risposta o, peggio, procedere alla cieca.

5. I protocolli di pagamento agentico: cosa sono, uno per uno, nel dettaglio

Un protocollo di pagamento agentico è un insieme di regole tecniche condivise che stabiliscono ESATTAMENTE come un agente AI deve dimostrare di avere il permesso di pagare, in un formato che qualunque negozio, banca o piattaforma può capire e verificare, senza che ognuno debba inventarsi da zero il proprio sistema.

Il settore è nato nel 2025 ed è ancora molto giovane: non esiste ancora UN SOLO standard universale accettato da tutti. Sono nati invece diversi protocolli, spinti da aziende diverse, ognuno con la propria filosofia e i propri sostenitori. TrustedPAI oggi ne supporta 4 su 5 di quelli rilevanti, proprio per poter dialogare con qualunque agente, indipendentemente da quale "lingua" tecnica parli: è un ruolo neutrale, non legato a nessuno dei protocolli in particolare.

Protocollo	Chi l'ha creato	Cosa risolve, in breve	Stato in TrustedPAI
AP2	Google + 60 organizzazioni partner	Mandati digitali firmati, prova crittografica del permesso dell'utente	Stabile
ACP	Stripe, OpenAI, Meta	"Buy it in ChatGPT": gettone di pagamento condiviso	Stabile
x402	Coinbase / x402 Foundation	Pagamenti istantanei in stablecoin dentro una richiesta web	Beta
MPP	Stripe + Tempo	Budget pre-autorizzato per micropagamenti a raffica in una sessione	Beta
UCP	Google + Shopify + 20 partner	Scoperta negozio/catalogo/carrello, non è pagamento in senso stretto	Non implementato

5.1 AP2: Agent Payments Protocol (Google)

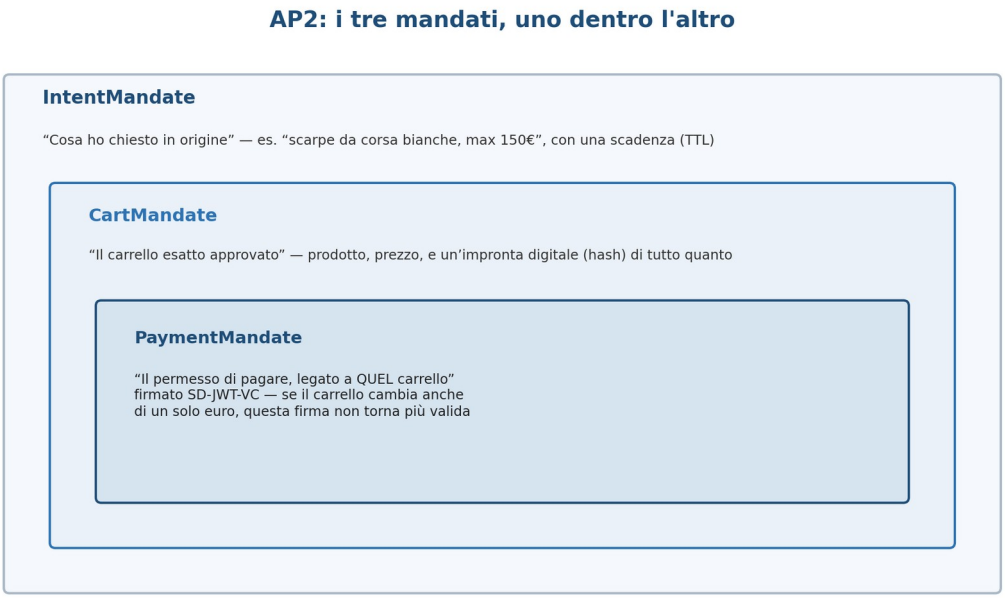
> **In sintesi, in una frase:** un sistema di tre permessi scritti e firmati, uno dentro l'altro, che dimostrano che un utente vero ha davvero autorizzato quell'acquisto esatto, a quel prezzo esatto.

Chi l'ha creato: Google, insieme a oltre 60 organizzazioni partner, tra cui PayPal, Mastercard, American Express, Coinbase, Etsy, Salesforce, Revolut, Adyen, ServiceNow, Intuit, MetaMask.

Il problema che risolve: prima di AP2 non esisteva un modo standard per un agente di dimostrare “l'utente vero mi ha davvero dato il permesso di comprare esattamente questo, a questo prezzo”. Ogni azienda avrebbe dovuto inventarsi il proprio sistema, incompatibile con quello di chiunque altro.

I tre mandati, nel dettaglio

Un “mandato” in AP2 è un contratto digitale firmato crittograficamente, impossibile da falsificare o modificare senza che la firma smetta di essere valida. Ce ne sono tre, uno dentro l'altro come scatole cinesi:



I tre mandati di AP2: ognuno vincolato a quello più interno.

Con l'esempio delle scarpe da corsa:

- **IntentMandate**: registra cosa l'utente ha chiesto in origine (“trovami delle scarpe da corsa bianche, non più di 150€”). Serve come traccia verificabile della richiesta iniziale, ed è particolarmente importante nello scenario “delegato”: contiene anche un limite di tempo (TTL, “time to live”) oltre il quale l'autorizzazione scade da sola, per evitare che resti valida all'infinito.
- **CartMandate**: il carrello esatto (prodotti, quantità, prezzo totale) che l'utente ha approvato, oppure che il negozio ha proposto e firmato per

essere poi approvato dall'agente secondo le regole dell'IntentMandate. Contiene un "hash" (un'impronta digitale matematica) di tutto il contenuto del carrello: se anche un solo numero nel prezzo cambiasse dopo la firma, l'impronta non corrisponderebbe più, e la manomissione salterebbe fuori subito.

- **PaymentMandate**: collega il carrello approvato al metodo di pagamento vero. È la parte più delicata dal punto di vista crittografico: usa una tecnologia chiamata SD-JWT-VC (una "credenziale digitale verificabile", uno standard in fase di definizione ufficiale, draft-ietf-oauth-sd-jwt-vc), estesa con un campo che lega indissolubilmente l'autorizzazione a QUEL CartMandate specifico. In pratica: anche se qualcuno rubasse questa autorizzazione, non potrebbe riusarla per pagare un carrello diverso: è cucita addosso a quella transazione e basta.

Come lo verifica davvero TrustedPAI

> ***Puoi saltare questo blocco se non ti interessa il dettaglio crittografico:*** *i prossimi punti spiegano ESATTAMENTE come TrustedPAI controlla le firme: utile se vuoi capire il meccanismo fino in fondo, non essenziale se ti basta sapere che "le firme vengono controllate a fondo, e i tentativi di imbroglio vengono scoperti" (puoi riprendere da "Esempio pratico" poco più sotto).*

- Controlla la firma del CartMandate (uno standard JWT, formato ES256) contro la chiave pubblica del negozio, registrata una sola volta in anticipo durante l'onboarding; mai scambiata dentro la singola richiesta di pagamento.
- Controlla la firma SD-JWT-VC del PaymentMandate contro la chiave pubblica dell'ente che ha emesso quella credenziale (il "credentials provider" di fiducia dell'utente).
- Ricalcola l'impronta del carrello e la confronta con quella firmata, per scoprire ogni tentativo di manomissione del prezzo dopo la firma.
- Tiene traccia di ogni identificatore di pagamento già usato, per bloccare i replay attack (vedi glossario).

Non esiste, ad oggi, un "ambiente di prova" pubblico per AP2 gestito da Google, a differenza di quanto Stripe offre per ACP (vedi sotto). Per questo motivo TrustedPAI genera internamente dati di test conformi alla specifica ufficiale, con

chiavi proprie, per verificare davvero tutto il meccanismo di firma senza dover dipendere da un servizio esterno che non esiste ancora.

Stato in TrustedPAI: stabile. Verificato con test reali contro 7 scenari di attacco diversi (importo manomesso dopo la firma, autorizzazione riusata su un carrello diverso, nonce sbagliato, chiavi false, replay), tutti correttamente bloccati.

> ***Approfondimento tecnico (facoltativo, livello avanzato):*** *la governance di AP2 è passata dalla sola Google alla FIDO Alliance, lo stesso consorzio dietro gli standard di autenticazione senza password (le "passkey"), un segnale che il settore lo tratta come un protocollo aperto, non come proprietà di un'unica azienda. Resta però un limite architetturale reale, condiviso da tutto l'ecosistema AP2 e non specifico di TrustedPAI: oggi ogni cliente può fidarsi di un solo "issuer" (l'ente che emette la credenziale di pagamento) alla volta: non esiste ancora un vero registro di fiducia multi-issuer. Una ricerca indipendente sulla sicurezza di AP2 conferma che è un gap riconosciuto a livello di settore, e suggerisce che una futura evoluzione potrebbe correlare anomalie TRA agenti e negozi diversi (non solo lo storico del singolo agente) per scoprire pattern di frode coordinata (i cosiddetti "slow cartel attack", dove più agenti diversi condividono lo stesso credentials provider o mostrano pattern geografici incoerenti).*

Esempio pratico: le scarpe da corsa, passo per passo

Per rendere concreto tutto questo, seguiamo lo scenario delle scarpe fino in fondo. L'utente dice al proprio assistente: "trovami scarpe da corsa bianche, non più di 150€". Ecco cosa succede dietro le quinte, in forma semplificata (non è codice reale, è una rappresentazione leggibile dei concetti):

```
IntentMandate:
  richiesta_originale: "scarpe da corsa bianche, max 150
EUR"
  utente: (identificativo firmato dell'utente)
  scade_il: 2026-08-25T00:00:00Z

CartMandate (dopo che l'agente ha trovato un paio a 129
EUR):
  prodotto: "Scarpe da corsa modello X, taglia 42, bianche"
  prezzo_totale: 129.00 EUR
  impronta_carrello: 7f3a9c... (hash di tutto quanto sopra)
  firma: (ES256, chiave del negozio)

PaymentMandate:
  collegato_a: impronta_carrello 7f3a9c...
```

```
  metodo_di_pagamento: (credenziale tokenizzata
dell'utente)
  firma: SD-JWT-VC (chiave dell'ente emittente della
credenziale)
```

A questo punto il negozio manda tutto questo a TrustedPAI (endpoint POST /v1/screen-transaction). TrustedPAI verifica prima le due firme e l'impronta del carrello: se anche solo il prezzo fosse stato alterato dopo la firma del CartMandate, l'impronta ricalcolata non corrisponderebbe e la transazione verrebbe bloccata all'istante, prima ancora di guardare le regole di rischio. Se le firme sono valide, TrustedPAI applica poi la policy del negozio (per esempio: “sotto i 200€, senza precedenti sull'agente, via libera automatico”) e restituisce APPROVE con un punteggio di rischio basso, insieme a una breve spiegazione. Solo a quel punto il negozio elabora il pagamento vero tramite il proprio processore, e l'ordine parte.

5.2 ACP: Agentic Commerce Protocol (Stripe, OpenAI, Meta)

> **In sintesi, in una frase:** un “buono” di pagamento usa-e-getta che permette all'agente di pagare tramite Stripe senza mai vedere il numero della carta vera.

Chi l'ha creato: nato dalla collaborazione tra Stripe e OpenAI, è il protocollo dietro “Buy it in ChatGPT” (Instant Checkout), e allargato in seguito con Meta come co-autore dello standard aperto.

Il problema che risolve: la stessa esigenza di AP2, ma con un approccio pensato per integrarsi rapidamente con l'infrastruttura di pagamento già esistente (in primis quella di Stripe, che processa già pagamenti per milioni di negozi).

I "pezzi" che compongono ACP

ACP non è un unico meccanismo, ma un insieme di componenti che si possono combinare:

- **Agentic checkout**: la sessione di acquisto vera e propria: creazione, aggiornamento e completamento del carrello, opzioni di consegna, elaborazione del pagamento.
- **Cart & feed**: un modo standard per un agente di sfogliare il catalogo prodotti di un negozio e costruire un carrello prima ancora di arrivare al pagamento.

- **Delegate payment**: il pezzo su cui si concentra TrustedPAI: il passaggio sicuro di un "gettone di pagamento" (Shared Payment Token, SPT) tra chi compra, l'agente, e il negozio, senza mai esporre i dati veri della carta all'agente. È come dare a qualcuno un buono spendibile una sola volta per un importo preciso, invece del numero della propria carta di credito.
- **Delegate authentication**: la delega dei permessi tramite OAuth 2.0 (uno standard molto diffuso, lo stesso usato ad esempio per "accedi con Google" su tanti siti), così l'agente può agire per conto dell'utente presso un negozio in modo tracciabile e revocabile.
- **Orders & webhooks**: notifiche automatiche sullo stato dell'ordine dopo il pagamento (conferma, spedizione, consegna, eventuale rimborso).

Come lo usa TrustedPAI: verifica lo Shared Payment Token contro la sandbox (ambiente di prova) reale che Stripe mette a disposizione, quindi non con dati inventati, ma contro l'infrastruttura vera di test di Stripe. Stato: stabile.

Esempio pratico: comprare dentro una chat

Immaginiamo un utente che chiede a un assistente AI integrato in un'app di chat "comprami questo zaino che mi hai appena mostrato". L'assistente, tramite ACP, ha già in mano uno Shared Payment Token, generato in precedenza quando l'utente ha collegato il proprio metodo di pagamento all'app, e valido per un solo utilizzo fino a un importo massimo concordato. L'assistente presenta questo token al negozio insieme al carrello (uno zaino, 79€). Il negozio inoltra tutto a TrustedPAI (endpoint POST /v1/screen-transaction-acp), che verifica il token contro l'infrastruttura di Stripe: è autentico? Non è già stato usato? È entro il limite concordato? Se tutto torna, e la policy del negozio lo consente, TrustedPAI risponde APPROVE e l'acquisto va a buon fine: l'utente non ha mai dovuto reinserire i dati della carta, e l'agente non li ha mai visti.

5.3 x402 (Coinbase / x402 Foundation)

> **In sintesi, in una frase:** il sito chiede il pagamento (con un vecchio codice web mai usato prima, il "402"), l'agente paga all'istante in criptovaluta stabile, e solo allora riceve quello che ha chiesto.

Chi l'ha creato: Coinbase, oggi passato sotto una fondazione più ampia (x402 Foundation), con il supporto di attori come AWS, Anthropic e Circle (l'azienda dietro la stablecoin USDC).

Il problema che risolve: pagamenti istantanei in stablecoin direttamente dentro una normale richiesta web, senza carte di credito, senza creare un account, senza gli attriti tipici dei pagamenti tradizionali online, pensato apposta per transazioni piccolissime e frequentissime, dove le commissioni delle carte di credito renderebbero l'operazione antieconomica.

Il flusso HTTP 402, passo per passo

x402 riusa un vecchio pezzo dimenticato di internet: il codice di errore HTTP "402 Payment Required", che esiste fin dagli anni '90 nello standard del web ma non era mai stato usato sul serio finora. Il flusso è questo:

```
1. Il client (un agente AI) chiede una risorsa a pagamento
a un server
2. Il server risponde: 402 Payment Required + prezzo e
valuta accettata
3. Il client firma un'istruzione di pagamento e la allega
alla richiesta
   (nell'header HTTP, non in un sistema separato)
4. La richiesta viene ripetuta, stavolta con il pagamento
incluso
5. Un "facilitator" verifica la firma e regola il pagamento
sulla blockchain
6. Il server consegna finalmente la risorsa richiesta
```

> ***Puoi saltare questo blocco se non ti interessa il dettaglio crittografico:*** *il prossimo paragrafo entra nel meccanismo di firma vero e proprio (EIP-3009, EIP-712, ECDSA): se ti basta sapere che "il pagamento viene autorizzato con una firma digitale sicura e regolato su una rete blockchain economica", puoi saltare al paragrafo "Come lo usa TrustedPAI".*

Il meccanismo di firma: x402 usa lo standard EIP-3009 ("transferWithAuthorization"), una funzione che permette di autorizzare un trasferimento di stablecoin con una firma EIP-712/ECDSA (lo stesso tipo di firma crittografica usata dai wallet digitali moderni) senza dover pagare separatamente una commissione di rete solo per "dare il permesso". Il pagamento viene poi regolato su una blockchain di "livello 2" (una rete costruita sopra Ethereum per essere molto più veloce ed economica, come Base), dettaglio tecnico che spiega perché x402 è diventato pratico solo di recente: su Ethereum "originale" le commissioni di rete avrebbero reso i micropagamenti antieconomici, sulle reti di livello 2 costano pochi centesimi.

Come lo usa TrustedPAI: verifica davvero la firma crittografica con la libreria ufficiale Python "x402" (non una reimplementazione fatta in casa), e fa anche una lettura diretta sulla blockchain (via RPC, "Remote Procedure Call", un modo per interrogare direttamente un nodo della rete) per controllare due cose che la sola firma non garantisce: che quel pagamento non sia già stato usato prima (anti-replay) e che ci siano davvero fondi sufficienti.

Stato: beta: copre solo lo schema "exact" su reti blockchain compatibili con Ethereum (Base, Base Sepolia, Polygon), non Solana, non Ethereum "principale", non altre reti annunciate ma non ancora coperte.

> ***Approfondimento tecnico (facoltativo, livello avanzato):*** *anche la governance di x402 è passata da Coinbase a un consorzio più ampio, oggi sotto l'ombrello della Linux Foundation, di nuovo un segnale di apertura verso l'intero settore, non il controllo di una sola azienda. Una nota architetturale interna interessante: quando in TrustedPAI si è iniziato a costruire il supporto a x402, l'ipotesi di partenza era che appartenesse alla stessa categoria di MPP: un protocollo "a sessione", con un budget aperto e speso nel tempo. L'implementazione reale ha smentito questa ipotesi: x402 si è rivelato appartenere alla stessa famiglia di AP2 e ACP (una firma autorizza esattamente un pagamento, non una sessione intera); è invece MPP, come si vede nella sezione 5.4, a comportarsi diversamente. È un promemoria utile: anche quando la documentazione ufficiale di un protocollo sembra chiara, l'unico modo per essere sicuri di come si comporta davvero è implementarlo e testarlo, non fermarsi alla lettura della specifica.*

Esempio pratico: un agente che paga per leggere un articolo

Un caso d'uso tipico di x402: un agente AI di ricerca prova ad aprire un articolo a pagamento per rispondere a una domanda dell'utente. Il server dell'editore risponde 402 Payment Required, indicando "0,02 USDC su Base". L'agente, che ha un piccolo wallet con fondi pre-caricati dall'utente proprio per questo scopo, firma un'autorizzazione EIP-3009 per quell'importo esatto e ripete la richiesta con la firma allegata. Prima che l'editore consegni l'articolo, la transazione passa da TrustedPAI (endpoint POST /v1/screen-transaction-x402), che controlla la firma, verifica su Base che quell'autorizzazione non sia già stata usata, e controlla che il wallet abbia davvero fondi sufficienti. Tutto questo avviene in una frazione di secondo, senza che l'utente debba fare nulla in quel momento: l'unico intervento umano è stato, prima, decidere quanto caricare sul wallet e per cosa poteva essere speso.

5.4 MPP: Machine Payments Protocol (Stripe & Tempo)

> **In sintesi, in una frase:** invece di autorizzare un pagamento alla volta, l'utente autorizza un budget massimo per una sessione intera, e l'agente spende dentro quel budget senza chiedere il permesso ogni singola volta.

Chi l'ha creato: nato a marzo 2026 dalla collaborazione tra Stripe e Tempo, una rete blockchain pensata apposta per pagamenti fatti da macchine.

Il problema che risolve: gli altri tre protocolli (AP2, ACP, x402) autorizzano un pagamento alla volta, ognuno con la propria firma. MPP pensa invece a uno scenario diverso: un agente che deve fare tanti micro-pagamenti di fila nello stesso contesto, per esempio un centesimo per ogni chiamata a un servizio, decine di volte al minuto. Chiedere una firma separata per ognuno sarebbe impraticabile: MPP fa autorizzare in anticipo un budget massimo per una "sessione", e poi l'agente spende dentro quel budget senza dover richiedere il permesso ogni singola volta.

Le tre modalità

> ***Puoi saltare questo blocco se non ti interessa il dettaglio crittografico:*** *le tre modalità differiscono soprattutto per dettagli tecnici di firma (voucher EIP-712, cifratura JWE): se ti interessa solo la logica generale ("un budget concordato in anticipo, speso a piccoli passi"), puoi scorrere fino a "Come lo usa TrustedPAI" poco più sotto.*

- **"stripe"**: concettualmente identica ad ACP: stesso meccanismo di Shared Payment Token.
- **"tempo"**: pagamenti a consumo su una rete blockchain dedicata (Tempo), con un sistema di "voucher" firmati (standard EIP-712, lo stesso di x402) che riportano un importo CUMULATIVO crescente: ogni nuovo voucher dichiara il totale speso fino a quel momento nella sessione, non solo l'ultimo pezzetto, così è sempre possibile verificare che il totale non abbia mai superato il budget concordato.
- **"card"**: un'estensione pensata per le carte di credito tradizionali (proposta da Visa): il payload di pagamento è cifrato (formato JWE) in modo che, per come è progettato il protocollo stesso, SOLO il PSP del negozio possa leggerlo davvero: non TrustedPAI, non chiunque altro. Per questo TrustedPAI, in questa modalità, fa solo controlli strutturali (il messaggio è nel formato giusto? non è già stato usato?) e dichiara sempre esplicitamente che la

verifica crittografica non è possibile; non è un limite di TrustedPAI, è un limite di design del protocollo stesso.

Come lo usa TrustedPAI: implementa tutte e tre le modalità. Quando la verifica crittografica non può essere certa (modalità “card”), il punteggio di rischio calcolato sale automaticamente: non viene mai restituito un via libera “pulito” in quel caso, proprio per non far sembrare più sicura una transazione di quanto lo sia davvero.

Stato: beta: la modalità “tempo” non legge ancora lo stato del canale di pagamento direttamente sulla blockchain (non esiste ancora un nodo RPC pubblico confermato per la rete Tempo), quindi si affida al voucher firmato più recente piuttosto che a una verifica on-chain indipendente.

> ***Approfondimento tecnico (facoltativo, livello avanzato):*** *internamente TrustedPAI classifica ogni protocollo in due famiglie architetturali: “mandato singolo” (una firma autorizza esattamente un pagamento: AP2, ACP, x402, vedi il box in fondo alla sezione 5.3) e “sessione streaming” (un'unica autorizzazione copre più micropagamenti nel tempo). MPP è oggi l'unico protocollo della seconda famiglia, ma solo in modo parziale, ed è onesto dirlo fino in fondo: ogni voucher viene ancora valutato come un'unità a sé quando arriva, non esiste ancora una vera “apertura di sessione” con una contabilità persistente del canale mantenuta da TrustedPAI tra un voucher e l'altro. Il controllo del budget funziona comunque correttamente (il cumulativo dichiarato in ogni voucher non può mai superare quanto concordato all'apertura), ma l'architettura pensata apposta per le sessioni streaming resta, ad oggi, un obiettivo solo parzialmente realizzato.*

Esempio pratico: una sessione di micropagamenti

Un agente AI che orchestra più strumenti a pagamento per completare un compito complesso (per esempio: cercare dati, tradurli, generare un'immagine, tutto tramite servizi diversi a consumo) apre una sessione MPP con un budget massimo di 5€ per l'intera sessione. Ogni volta che l'agente usa un servizio, viene generato un nuovo voucher firmato con il totale cumulativo speso finora (0,10€, poi 0,35€, poi 0,80€…). Ogni voucher, prima di essere accettato dal servizio che lo riceve, passa da TrustedPAI (endpoint POST /v1/screen-transaction-mpp), che verifica la firma e controlla che il cumulativo dichiarato non superi mai il budget dei 5€ concordato all'apertura della sessione: se un voucher dichiarasse un cumulativo superiore al budget, verrebbe rifiutato immediatamente, anche se la firma fosse perfettamente valida.

5.5 UCP: Universal Commerce Protocol (Google, Shopify e altri), non ancora implementato

> **In sintesi, in una frase:** non è un modo per pagare, è un modo per un negozio di "farsi trovare e capire" da un agente AI: per pagare, userà comunque uno degli altri quattro protocolli.

Chi l'ha creato: Google insieme a Shopify, con l'adesione di oltre 20 partner tra cui Etsy, Walmart, Target, Stripe, Visa, Mastercard.

Cosa fa, e perché è diverso dagli altri quattro: AP2, ACP, x402 e MPP si occupano SOLO della parte finale: autorizzare il pagamento. UCP invece copre l'intero percorso di acquisto: come un negozio si fa "scoprire" dagli agenti AI (Discovery), come mostra il proprio catalogo prodotti in un formato standard (Catalog), come gestisce il carrello e il checkout (Cart & Checkout), e come gestisce identità/pagamento tokenizzato con prova crittografica del consenso. Nasce per risolvere quello che Google chiama il "problema N×N": senza uno standard comune, ogni negozio dovrebbe integrarsi separatamente con ogni singola piattaforma AI (Google, OpenAI, ecc.), moltiplicando il lavoro. UCP è pensato per essere compatibile con AP2 proprio sulla parte di pagamento: UCP gestisce "il negozio", AP2 gestisce "la cassa".

Perché non è ancora in TrustedPAI: è l'unico dei 5 protocolli rilevanti che manca oggi. Non essendo un protocollo di autorizzazione di pagamento in senso stretto (più simile a "come si struttura un negozio online per essere visitabile da un agente"), la sua rilevanza diretta per un motore di rischio come TrustedPAI dipende da come evolverà nei prossimi mesi; resta nella lista delle cose da valutare, non è stato scartato.

6. Un esempio completo, dall'inizio alla fine

Le sezioni precedenti hanno spiegato ogni pezzo separatamente. Questa sezione li rimette insieme in un'unica storia continua, per vedere come si comportano DAVVERO in sequenza, con un caso concreto e credibile.

Marco usa un assistente AI generico a cui ha dato accesso a un metodo di pagamento, tramite AP2, con un limite di spesa mensile di 300€ concordato in anticipo con l'assistente stesso. Un giorno dice: “se il volo Milano-Barcellona scende sotto i 90€ nelle prossime due settimane, prenotalo, altrimenti aspetta”.

6. L'assistente registra la richiesta come IntentMandate: destinazione, prezzo massimo, finestra temporale di due settimane, TTL corrispondente.
7. Dieci giorni dopo, l'assistente trova un volo a 84€ e costruisce il CartMandate: compagnia aerea, data, prezzo, con l'impronta digitale del carrello e la firma del negozio (in questo caso, l'agenzia di viaggi online).
8. L'assistente genera il PaymentMandate, collegato a quel CartMandate esatto, firmato con la credenziale di pagamento di Marco (SD-JWT-VC).
9. L'agenzia di viaggi, prima di confermare la prenotazione, chiama TrustedPAI su POST /v1/screen-transaction con tutti e tre i mandati e l'importo (84€).
10. TrustedPAI verifica le firme: il CartMandate è autentico e non manomesso (l'impronta ricalcolata corrisponde), il PaymentMandate è autentico e legato esattamente a quel carrello, nessuno dei due identificatori è già stato usato in passato.
11. TrustedPAI applica la policy dell'agenzia di viaggi: “sotto i 500€, con agente senza precedenti, presenza umana non richiesta sotto i 200€”, 84€ è ben dentro questi limiti, e lo storico dell'agente di Marco (interrogabile su GET /v1/agents/{id}/history) non ha alcun precedente negativo.
12. Verdetto: APPROVE, punteggio di rischio molto basso (per esempio 4 su 100), motivazione: “firme valide, importo entro policy, nessun precedente sull'agente”.
13. L'agenzia di viaggi, ricevuto il via libera, elabora il pagamento vero tramite il proprio processore e conferma la prenotazione. Marco riceve la notifica solo a cose fatte: “ho prenotato il volo a 84€ come mi avevi chiesto”.

Da notare: in questo intero flusso, TrustedPAI non ha mai visto il numero di una carta, non ha mai mosso un euro, e ha preso una decisione in una manciata di millisecondi: il tempo che a un umano servirebbe anche solo per aprire l'email di conferma.

7. Due casi limite raccontati per intero: un HOLD e un REJECT

Sapere che esistono tre verdetti possibili è una cosa; vedere DAVVERO cosa succede quando non arriva un APPROVE pulito è un'altra. Ecco due scenari concreti, uno per ciascun caso.

7.1 Un caso di HOLD: importo alto, nessuna conferma di presenza umana

Stesso Marco, stesso assistente, ma stavolta la richiesta iniziale era più vaga: “se trovi un buon pacchetto vacanza entro l'estate, prenotalo pure”. Mesi dopo, l'assistente trova un pacchetto a 1.450€ e lo prenota autonomamente, componendo i tre mandati AP2 esattamente come nell'esempio precedente: le firme sono tutte perfettamente valide, nessuna manomissione, nessun replay.

Il negozio manda tutto a TrustedPAI. Le firme passano il controllo crittografico senza problemi. Ma la policy del negozio dice: “sopra i 1.000€, richiedi conferma esplicita di presenza umana nel PaymentMandate, altrimenti HOLD”, e in questo caso quel campo non è presente, perché la richiesta originale di Marco era genuinamente delegata, senza conferma nel momento dell'acquisto.

Verdetto: HOLD, punteggio di rischio moderato (per esempio 58 su 100), motivazione: “importo sopra soglia di 1.000€ senza conferma di presenza umana, richiede revisione manuale prima di procedere”. Il negozio, a questo punto, può scegliere di contattare Marco per una conferma esplicita prima di finalizzare, oppure lasciare la prenotazione in sospeso fino a una decisione umana da parte sua. TrustedPAI non ha bloccato la transazione in modo definitivo: ha segnalato che serve un controllo in più, senza dare né un sì né un no automatico.

7.2 Un caso di REJECT: importo manomesso lungo il tragitto

Un altro agente, di un altro utente, costruisce un CartMandate per un ordine di elettronica da 45€, correttamente firmato dal negozio. Tra il momento della firma e il momento in cui la richiesta arriva effettivamente al negozio, qualcosa nel percorso (un componente compromesso, un tentativo di manomissione) altera il campo del prezzo totale a 450€, sperando che il controllo si limiti a leggere il numero senza riverificarlo.

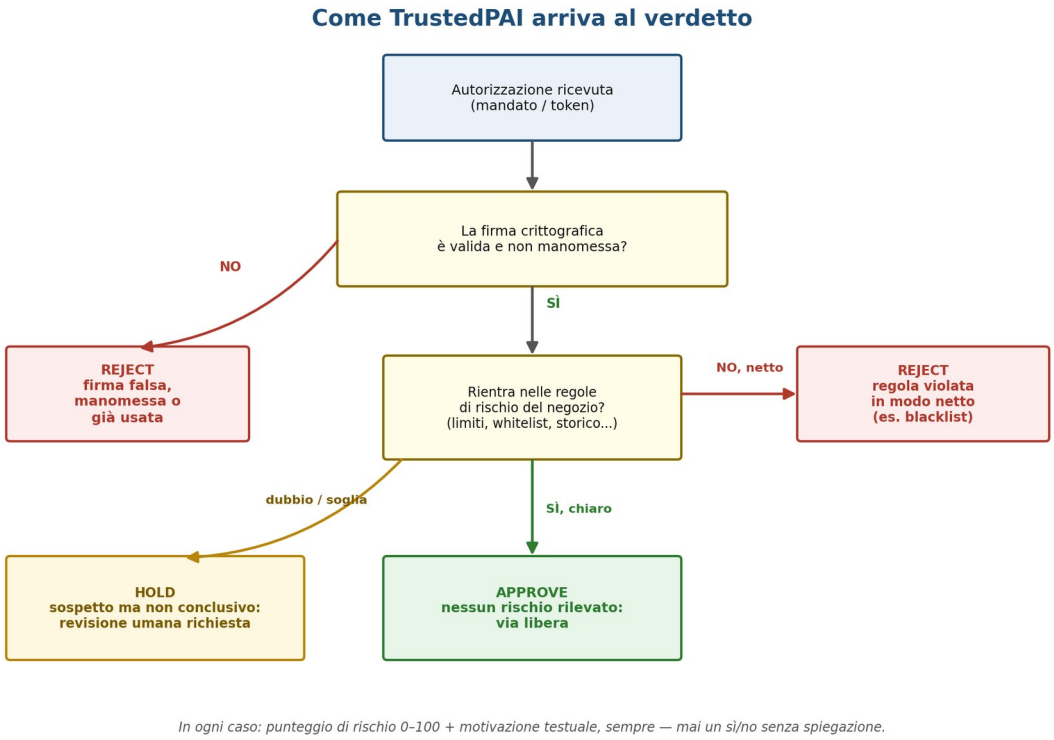
TrustedPAI, come fa sempre, ricalcola l'impronta digitale dell'intero carrello a partire dai dati ricevuti e la confronta con quella firmata al momento della

creazione del CartMandate. Le due impronte non corrispondono più: è la prova matematica che qualcosa è stato alterato dopo la firma, chiunque sia stato.

Verdetto: REJECT, motivazione esplicita: “impronta del carrello non corrispondente alla firma originale, possibile manomissione, transazione bloccata”. Non c'è margine di ambiguità in un caso così: non viene proposto un HOLD per una revisione umana, perché la prova crittografica di manomissione è già conclusiva di per sé: un umano non avrebbe comunque modo di “validare” un'impronta digitale che non torna, quindi non ha senso far perdere tempo a nessuno con una revisione manuale in un caso come questo.

8. Il motore di rischio: come TrustedPAI decide APPROVE / HOLD / REJECT

Verificare che una firma sia autentica è solo il primo livello di controllo, ed è un controllo binario (o è valida, o non lo è). Il secondo livello, indipendente dalla crittografia, è dove TrustedPAI applica un vero giudizio di rischio, basato sulle regole che il singolo negozio ha configurato per il proprio account. Ogni negozio ha una propria policy, diversa da quella di un altro negozio: non esiste un'unica regola valida per tutti. Il diagramma qui sotto riassume tutta la logica in un colpo d'occhio; i paragrafi che seguono la spiegano nel dettaglio.



Dalla firma crittografica alle regole di rischio, fino al verdetto finale.

Le dimensioni di rischio valutate

- **Limite massimo per singola transazione**: es. “non approvare mai automaticamente ordini sopra i 2.000€”: oltre quella soglia, il verdetto tende verso HOLD anche se la firma è perfetta.
- **Limite massimo giornaliero per lo stesso agente**: per accorgersi se un agente sta comprando in modo anomalo o eccessivo in un lasso di tempo breve, sommando tutte le sue transazioni della giornata.

- **Whitelist / blacklist**: elenchi espliciti di merchant/destinatari sempre approvati o sempre bloccati, a prescindere dal resto.
- **Soglia di presenza umana**: sopra un certo importo, il negozio può richiedere che sia stato dichiarato esplicitamente che una persona vera era presente e ha confermato l'acquisto in quel momento, non solo l'agente da solo (rilevante soprattutto nello scenario “delegato” descritto nella sezione 2, e visto in azione nell'esempio 7.1).
- **Storico e reputazione dell'agente**: se quello stesso identificativo di agente ha già avuto segnalazioni di abuso confermate in passato (vedi sezione 10.3), il punteggio di rischio calcolato per le transazioni successive sale automaticamente: un agente “con precedenti” parte da una soglia di attenzione più alta.
- **Affidabilità della verifica crittografica stessa**: non tutti i protocolli/modalità danno la stessa certezza matematica (vedi MPP/“card” nella sezione 5.4): quando la verifica crittografica non può essere completa al 100%, questo pesa a favore di un rischio più alto, mai il contrario.

I tre verdetti possibili

- **APPROVE**: nessun rischio rilevato secondo la policy configurata per quel negozio: procedi pure. Esempio visto in sezione 6.
- **HOLD**: sospetto ma non conclusivo: serve una persona che lo controlli a mano prima di procedere. Esempi tipici: importo alto senza conferma di presenza umana (visto in sezione 7.1), agente con storico di precedenti rischiosi, verifica crittografica non completa per limiti del protocollo/modalità usata.
- **REJECT**: blocca del tutto: o la firma crittografica non torna (mandato falso, manomesso, visto in sezione 7.2, o già usato in precedenza), oppure la policy del negozio esclude esplicitamente questo caso (es. destinatario in blacklist).

> ***Approfondimento tecnico (facoltativo, livello avanzato):*** *sotto il cofano, ogni protocollo si collega al motore di rischio tramite un contratto software comune (un'interfaccia interna chiamata PaymentProtocol), che espone un solo metodo (“verifica e normalizza”) capace di trasformare il formato specifico di ciascun protocollo (mandati AP2, token ACP, firme x402, voucher MPP) in una struttura dati comune che il motore di rischio può giudicare senza sapere da quale protocollo arriva. Questa struttura comune porta con sé un campo booleano esplicito, “verificato*

> *crittograficamente": quando è falso (per esempio nella modalità "card" di MPP, strutturalmente non verificabile, sezione 5.4) una regola dedicata del motore di rischio alza automaticamente il punteggio calcolato. È il meccanismo tecnico esatto dietro l'affermazione, ripetuta più volte in questo documento, che TrustedPAI non fa mai finta di una certezza crittografica che non ha davvero.*

Ogni verdetto è accompagnato da un punteggio di rischio numerico da 0 a 100 e da una o più spiegazioni testuali del perché: mai un semplice "sì" o "no" senza motivazione, anche quando il verdetto è APPROVE.

Come già anticipato nella sezione 4: se qualcosa va storto internamente a TrustedPAI (non nella transazione, proprio nel servizio), il sistema non lascia mai il negozio senza risposta: restituisce comunque un HOLD prudente con motivazione esplicita, mai un errore silenzioso e mai un'approvazione automatica per pigrizia del sistema in un momento di difficoltà tecnica.

9. Gli endpoint: cosa si può effettivamente chiedere a TrustedPAI

Un “endpoint” è semplicemente un indirizzo specifico dell'API a cui un programma può mandare una richiesta per ottenere un certo servizio. Ecco l'elenco completo di quelli disponibili oggi, spiegati in parole semplici:

Endpoint	Cosa fa
POST /v1/screen-transaction	Valuta una transazione autorizzata via AP2 (la catena Intent/Cart/PaymentMandate).
POST /v1/screen-transaction-acp	Valuta una transazione autorizzata via ACP (Shared Payment Token di Stripe).
POST /v1/screen-transaction-x402	Valuta una transazione autorizzata via x402 (pagamento in stablecoin on-chain).
POST /v1/screen-transaction-mpp	Valuta una transazione autorizzata via MPP, in una delle tre modalità (stripe/tempo/card).
GET /v1/health	Controllo di base: il servizio è raggiungibile e funzionante? Non richiede autenticazione, utile prima di integrare gli altri endpoint.
GET /v1/agents/{id}/history	Restituisce lo storico e la reputazione di un agente specifico: quante transazioni, quante bloccate, eventuali abusi confermati.
PATCH /v1/policy	Permette al negozio di modificare da solo (self-service) le proprie regole di rischio, senza dover chiedere un intervento manuale.
POST /v1/report-abuse	Permette al negozio di segnalare, dopo il fatto, che una transazione già avvenuta si è rivelata un abuso (reso fraudolento, rimborso sospetto, rivendita, chargeback ingiustificato, altro).
GET /v1/my-transactions	Restituisce le transazioni del negozio che chiama, con totali ed esiti, la base della dashboard cliente.
GET /admin/dashboard	Pannello di monitoraggio interno (uso di chi gestisce TrustedPAI, non del singolo cliente), protetto da un token di accesso: grafico andamento, elenco transazioni, filtri di ricerca.

Tutti gli endpoint (tranne /v1/health) richiedono una chiave API assegnata al singolo cliente nell'header della richiesta (“X-API-Key”). Una chiave mancante, sbagliata, o disattivata restituisce sempre un errore di autenticazione prima ancora che il corpo della richiesta venga letto.

Oltre alla via REST classica, tutte queste funzioni sono esposte anche tramite un server MCP (Model Context Protocol, uno standard creato da Anthropic; vedi sezione 10.5), così un agente AI può “chiamare” TrustedPAI come farebbe con uno strumento della propria cassetta degli attrezzi, senza passare da una chiamata web tradizionale.

10. Le funzionalità oltre la pura verifica (l'“arsenale competitivo”)

Valutare una transazione isolata, una alla volta, non basta per essere competitivi in questo settore: lo dimostra il fatto che i principali concorrenti (aziende come Forter, Riskified, Signifyd, Visa) offrono tutti qualcosa di più di una semplice verifica puntuale. Per questo motivo TrustedPAI include cinque funzionalità aggiuntive, costruite riusando sempre lo stesso motore di rischio, senza duplicare logica:

10.1 Storico e reputazione dell'agente nel tempo

TrustedPAI non giudica solo la singola transazione isolata: tiene traccia di cosa ha fatto quello stesso agente in passato (quante transazioni totali, quante bloccate, eventuali abusi confermati) e usa questo storico per pesare il rischio delle transazioni successive. Un dettaglio onesto da sapere: oggi questo storico è condiviso tra TUTTI i clienti di TrustedPAI per lo stesso identificativo di agente, non isolato per singolo negozio: se un agente ha avuto un precedente presso il Negozio A, quel precedente pesa anche quando lo stesso agente si presenta al Negozio B. È di fatto una prima forma, ancora semplice, di rete di reputazione condivisa: un agente che si comporta male non “riparte pulito” semplicemente cambiando negozio.

10.2 Policy builder self-service

Il negozio può cambiare le proprie regole di rischio (limiti di spesa, whitelist/blacklist, soglie di presenza umana) da solo, in autonomia, tramite l'endpoint PATCH /v1/policy, senza dover chiedere una modifica manuale al team di TrustedPAI. La modifica si applica davvero al motore di rischio in tempo reale, non solo alla risposta di conferma: la transazione successiva, anche a pochi secondi di distanza, viene già valutata con la nuova regola.

10.3 Gestione degli abusi dopo il pagamento

Un pagamento può sembrare pulito al momento della verifica, e rivelarsi un abuso solo dopo: un reso fraudolento, un rimborso sospetto, una rivendita non autorizzata, un chargeback ingiustificato. Il negozio può segnalarlo esplicitamente a TrustedPAI tramite POST /v1/report-abuse, indicando la categoria; TrustedPAI ne tiene conto per le valutazioni future dello stesso agente (vedi 10.1). Un controllo di sicurezza importante: un negozio non può segnalare un abuso su una transazione che non è sua: il sistema verifica sempre questo

isolamento, per evitare che un negozio possa danneggiare la reputazione di un agente su una transazione che non ha nemmeno gestito.

10.4 Dashboard per il negozio

Tramite GET /v1/my-transactions, ogni negozio può vedere le proprie transazioni, quante approvate/bloccate/in revisione, senza dover leggere righe di database a mano o chiedere un report manuale. È distinta dalla dashboard amministrativa interna (sezione 9), che invece è ad uso di chi gestisce TrustedPAI stesso e mostra dati aggregati su tutti i clienti.

10.5 Un server MCP

MCP (Model Context Protocol) è uno standard aperto creato da Anthropic per far “parlare” un assistente AI con strumenti esterni in modo strutturato, senza dover costruire un'integrazione su misura ogni volta. TrustedPAI espone un proprio server MCP che mette a disposizione, come “strumenti” utilizzabili direttamente da un agente AI, le stesse funzioni di screening, di consultazione dello storico agente e di segnalazione abuso disponibili via REST, usando esattamente la stessa logica interna (nessuna duplicazione, nessun rischio che le due vie diano risposte diverse per lo stesso caso).

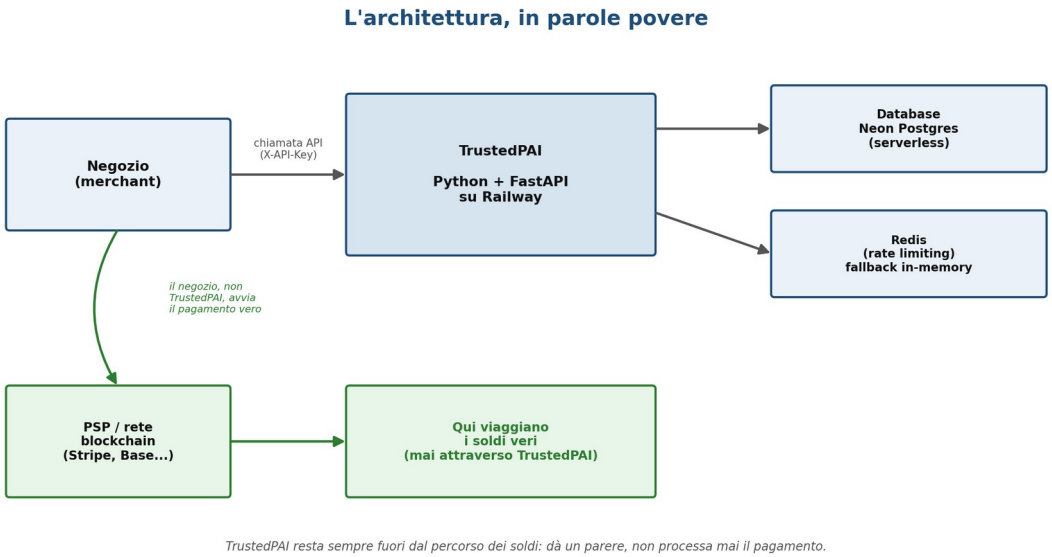
11. Sicurezza applicata e architettura tecnica, in parole semplici

11.1 Sicurezza

- **Rate limiting**: ogni chiave API può fare al massimo 60 richieste al minuto. Il controllo avviene PRIMA della verifica della chiave stessa, apposta: così nemmeno un tentativo con una chiave sbagliata/inventata può essere usato per “provare a indovinare” chiavi valide a raffica.
- **Anti-replay**: per ogni protocollo, TrustedPAI tiene traccia degli identificatori di autorizzazione già visti (token ACP, identificatore del PaymentMandate AP2, coppia mittente+nonce per x402, canale+importo cumulativo per MPP/tempo) per bloccare ogni tentativo di riutilizzo.
- **Cifratura delle chiavi dei clienti**: le chiavi segrete Stripe registrate dai clienti sono cifrate nel database (algoritmo Fernet), mai salvate in chiaro; non viaggiano mai nel corpo di una singola richiesta di screening.
- **Fail-safe, non fail-open**: come già detto nella sezione 4 e 8: un errore interno produce sempre un HOLD prudente, mai un'approvazione automatica per default.
- **Monitoraggio errori**: un sistema di error tracking (Sentry) segnala automaticamente quando qualcosa si rompe nel codice, escludendo esplicitamente i dati di pagamento veri e propri da quanto viene inviato a quel servizio esterno.

11.2 Stack tecnico

Un modo semplice per vedere come si incastrano i pezzi tecnici tra loro:



Il negozio parla con TrustedPAI via API; i soldi veri passano sempre da un'altra strada.

- **Linguaggio e framework**: Python, con FastAPI (uno dei framework più diffusi per costruire servizi web che rispondono velocemente).
- **Hosting**: Railway, un servizio che ospita ed esegue il codice: non serve gestire server fisici o configurazioni infrastrutturali complesse.
- **Database**: Neon, un Postgres “serverless” (si adatta da solo al carico, costa poco quando poco usato), oggi su piano gratuito, con il limite che può sospendersi temporaneamente per inattività prolungata.
- **Coda/limite richieste**: Redis (un database in memoria molto veloce), usato per il rate limiting condiviso tra le varie copie del servizio in esecuzione; se Redis non è disponibile, il sistema passa automaticamente a un conteggio interno più semplice, invece di smettere di funzionare del tutto.

11.3 Come viene testato

Ogni protocollo ha una suite di test automatici che gira contro un server vero e un database vero (non simulazioni finte): controlla sia i casi “puliti” (transazione legittima approvata correttamente) sia scenari di attacco (firma falsa, importo manomesso, autorizzazione riusata, agente con storico di abusi). Per le parti che richiedono servizi esterni reali difficili da testare in automatico (es. la sandbox vera di Stripe, un nodo blockchain pubblico), quei singoli pezzi esterni vengono sostituiti con controlli equivalenti in un ambiente di test controllato: mai la logica di verifica crittografica stessa, che è sempre quella vera.

12. Cosa TrustedPAI NON fa (per essere chiari fino in fondo)

- Non gestisce mai i soldi direttamente: non è un processore di pagamento, non tocca mai le carte di credito o i wallet.
- Non crea né possiede gli agenti AI: non è OpenAI, non è Google, è indipendente da chi costruisce l'agente.
- Non decide al posto del negozio: dà un consiglio (APPROVE / HOLD / REJECT), ma la decisione finale di procedere resta sempre del negozio o del suo PSP.
- Non garantisce mai una verifica crittografica al 100% in ogni modalità: quando un protocollo stesso (es. MPP/“card”) non la permette per come è progettato, TrustedPAI lo dichiara esplicitamente invece di far finta.
- Non è ancora un prodotto “enterprise-ready” in senso stretto: vedi la sezione 14 sui limiti onesti.

13. Domande frequenti

Una raccolta delle domande più naturali che vengono in mente leggendo questo documento per la prima volta, con risposte dirette.

TrustedPAI può rubare o trattenere i miei soldi?

No. TrustedPAI non tocca mai i soldi in nessuna fase: non li riceve, non li trattiene, non li trasferisce. Il pagamento vero e proprio avviene sempre e solo tramite un processore di pagamento reale (Stripe, una rete blockchain). TrustedPAI si limita a dare un giudizio PRIMA che quel pagamento venga eseguito da qualcun altro.

Cosa succede se l'agente AI sbaglia e compra qualcosa che non volevo?

Dipende da dove si trova l'errore. Se la richiesta originale era ambigua e l'agente ha interpretato male (sezione 2, categoria “errore in buona fede”), TrustedPAI non può saperlo: le firme sono valide, la transazione rientra nella policy, e il verdetto sarà APPROVE, perché dal punto di vista di TrustedPAI tutto è regolare. Questo è un limite reale e onesto: TrustedPAI verifica che l'AUTORIZZAZIONE sia autentica e rientri nelle regole di rischio, non può leggere nella testa dell'utente cosa intendeva davvero dire. Le protezioni contro questo tipo di errore stanno a monte, nel modo in cui l'utente formula le istruzioni e nei limiti di spesa che imposta sull'agente stesso.

TrustedPAI sa chi sono io, come persona?

TrustedPAI riceve gli identificativi già presenti nei mandati/token di pagamento (es. un identificativo dell'agente, un riferimento al metodo di pagamento tokenizzato), non un profilo della persona in senso ampio. Non è un sistema di identità: si appoggia alle credenziali già verificate da altri (l'ente emittente della credenziale in AP2, Stripe in ACP) piuttosto che costruire una propria anagrafica degli utenti finali.

Chi paga se TrustedPAI si sbaglia e approva una transazione che poi si rivela una frode?

Oggi, senza un contratto di servizio (SLA) formale né un'assicurazione di responsabilità professionale (vedi sezione 14.1), la responsabilità finanziaria di una decisione sbagliata resta in capo al negozio che ha scelto di fidarsi del verdetto, esattamente come oggi un sistema antifrode tradizionale dà un punteggio, ma la decisione finale e la responsabilità restano del negozio che lo

usa. È uno dei motivi per cui, nella sezione 15, l'idea di garanzie assicurate sulla decisione di rischio è indicata come uno sviluppo futuro, non ancora presente.

Perché non basta usare Stripe Radar o un sistema antifrode tradizionale già esistente?

I sistemi antifrode tradizionali sono ottimizzati per riconoscere comportamento umano anomalo: pattern di digitazione, movimento del mouse, cronologia di navigazione, dispositivo usato. Un agente AI non genera questi segnali, o li genera in modo del tutto diverso da un umano, quindi quei modelli non sono automaticamente efficaci. TrustedPAI, al contrario, è costruito attorno ai segnali specifici del commercio agentico: la validità crittografica di un mandato firmato, lo storico di un identificativo di agente, la conformità a un protocollo come AP2/ACP/x402/MPP. Sono complementari, non sostituti l'uno dell'altro.

TrustedPAI funziona solo se è coinvolta un'intelligenza artificiale?

Sì, nel senso che è pensato specificamente per il caso in cui chi autorizza il pagamento è un agente che agisce per conto di una persona, tramite uno dei protocolli supportati. Un pagamento fatto direttamente da un umano su un sito normale, senza nessun agente di mezzo, non passa da nessuno dei quattro endpoint di screening: non è il caso d'uso per cui esiste.

Cosa succede se domani esce un protocollo nuovo, il sesto?

Tecnicamente, aggiungere un nuovo protocollo significa costruire un nuovo endpoint di verifica specifico per quel formato di firma/mandato, riusando lo stesso motore di rischio già esistente (le sezioni 8 e 10 non cambiano). È esattamente il percorso già seguito quattro volte (AP2, ACP, x402, MPP) e, in prospettiva, il quinto per UCP quando/se sarà rilevante (sezione 5.5). L'architettura è stata pensata fin dall'inizio per essere neutrale rispetto al protocollo, proprio per questo motivo.

Un negozio può ignorare il verdetto di TrustedPAI e procedere comunque?

Sì. TrustedPAI dà un consiglio, non ha alcun potere di bloccare tecnicamente un pagamento: non è lui a eseguirlo. Un negozio potrebbe, in teoria, ricevere un REJECT e processare comunque il pagamento tramite il proprio processore. Sarebbe una scelta del negozio che vanifica lo scopo del servizio, ma tecnicamente resta possibile: TrustedPAI è un consulente, non un guardiano con potere di veto assoluto sul flusso di denaro.

È sicuro affidare le chiavi segrete di Stripe a TrustedPAI?

Le chiavi vengono cifrate nel database con l'algoritmo Fernet (sezione 11.1) e non vengono mai incluse nel corpo delle singole richieste di screening. Detto questo, come indicato onestamente nella sezione 14, non esiste ancora un audit di sicurezza indipendente né una certificazione (SOC 2, ISO 27001) che lo confermi da una terza parte esterna: oggi la garanzia si basa sulla descrizione onesta di come funziona il codice, non su una verifica esterna certificata.

TrustedPAI è open source?

I protocolli che TrustedPAI implementa (AP2, ACP, x402, MPP) sono standard aperti e pubblici, consultabili da chiunque. Il codice specifico di TrustedPAI (il motore di rischio, le policy, l'infrastruttura) non è reso pubblico oggi.

Come guadagna TrustedPAI, concretamente?

Il modello discusso, descritto per esteso nella sezione 15, prevede che paghi chi riceve il pagamento (il negozio o il suo PSP), non chi possiede l'agente né l'infrastruttura di pagamento sottostante. Il percorso concreto verso il primo incasso reale (un pilota, poi un modello a consumo) è delineato ma non ancora avviato, per scelta deliberata (prima completare il prodotto).

Cosa succede se il database di TrustedPAI si rompe proprio mentre sta valutando una transazione?

Come spiegato nella sezione 4 e ribadito nella sezione 8: il sistema è progettato per rispondere comunque, con un HOLD esplicito e motivato (“valutazione automatica fallita, trattenuto per sicurezza”), non con un errore muto né con un'approvazione automatica di ripiego. È una scelta di design intenzionale, chiamata “fail-safe”: in caso di dubbio o di guasto, il sistema pende sempre verso la cautela, mai verso il lasciar correre.

Cosa vuol dire davvero “beta” per x402 e MPP?

Vuol dire che la verifica crittografica di base funziona ed è stata testata, ma la copertura non è ancora completa quanto quella di AP2 e ACP (“stabili”): per x402 mancano reti come Solana o Ethereum “principale”; per MPP la modalità “tempo” non legge ancora lo stato del canale direttamente sulla blockchain (sezione 5.4). Non significa “non funzionante”, significa “copertura parziale, dichiarata esplicitamente come tale”.

Chi controlla che TrustedPAI stesso sia onesto e sicuro?

Oggi, onestamente, nessuna terza parte esterna lo ha ancora certificato: è uno dei punti più importanti della sezione 14. Non c'è ancora un audit indipendente, non c'è una certificazione di settore. È una delle ragioni per cui questo stesso documento include, deliberatamente, un intero capitolo sui limiti reali del progetto invece di raccontare solo la parte migliore.

Il progetto si ferma qui, con quello che è scritto in questo libro?

No, anzi, è quasi certo che diversi dettagli qui dentro cambieranno nei prossimi mesi, dato quanto velocemente si sta muovendo tutto il settore del commercio agentico (sezione 3). Questo documento fotografa lo stato del progetto e dei protocolli così come sono a metà del 2026: è un buon punto di partenza per capire la logica di fondo, non un testo che resterà valido in ogni dettaglio per sempre.

14. Stato onesto del progetto: cosa c'è oggi e cosa manca ancora

Questa sezione esiste apposta per non raccontarsi solo la parte migliore. È divisa per area.

14.1 Legale e societario

- Nessuna entità legale costituita (oggi è un progetto di una persona fisica, non una società).
- Nessun ToS (termini di servizio) o SLA (accordo sui livelli di servizio) formali.
- Nessun DPA (accordo sul trattamento dati, rilevante per il GDPR europeo).
- Nessuna assicurazione di responsabilità civile professionale.

14.2 Fiducia tecnica

- Nessun audit di sicurezza indipendente ancora effettuato.
- Nessuna certificazione (es. SOC 2, ISO 27001) ottenuta.
- Database su piano gratuito Neon, che può sospendersi per inattività prolungata.
- Cifratura dei dati transazionali “a riposo” (oltre alle chiavi Stripe, già cifrate) valutata e deliberatamente rimandata: il database sottostante è già cifrato a livello di piattaforma da Neon, e cifrare anche a livello applicativo romperebbe la ricerca testuale della dashboard senza un beneficio di sicurezza chiaro: una scelta esplicita, non una dimenticanza.

14.3 Affidabilità operativa

- Nessun monitoraggio in tempo reale di anomalie di rischio o di tempi di fermo del servizio (uptime): solo tracciamento degli errori applicativi.
- Nessun alert automatico (es. su Slack o email) in caso di problema.
- Nessuna SLA di disponibilità dichiarata.
- Singolo punto di guasto sul database: nessun sistema di ripristino in caso di disastro (disaster recovery) ancora testato.

14.4 Business

- Nessun cliente pilota reale ancora avviato; decisione deliberata: prima completare le funzionalità di cui sopra, poi cercare un primo cliente.
- Nessuna fatturazione automatica a consumo.
- Nessun SDK dedicato per chi costruisce agenti AI (es. per framework come LangChain).
- Nessun sito ufficiale ancora: il dominio trustedpai.com è posseduto ma un sito vero verrà costruito solo a prodotto maturo, pronto per essere presentato e venduto.

14.5 Copertura tecnica

- 4 protocolli di pagamento su 5 rilevanti coperti: manca solo UCP (vedi sezione 5.5).
- AP2 supporta un solo “issuer” (ente emittente di credenziali) di fiducia per cliente: un vero “registro di fiducia” multi-issuer non esiste ancora.
- x402 limitato a reti blockchain compatibili con Ethereum (no Solana, no Ethereum “principale”).
- MPP/“tempo” non legge ancora lo stato del canale di pagamento direttamente sulla blockchain.
- Onboarding di un nuovo cliente ancora manuale (fatto a mano); solo la modifica delle regole di rischio, una volta che l'account esiste, è già self-service.

14.6 Cosa servirebbe per vendere ad aziende grandi/enterprise

Un elenco più esteso, pensato specificamente per il tipo di clienti più esigenti (banche, grandi retailer, aziende regolamentate), deliberatamente non perseguito finché non arriva una richiesta concreta da un cliente reale, per non investire tempo in cose che nessuno ha ancora chiesto:

- Certificazione SOC 2 Type II, penetration test periodici, una policy pubblica di divulgazione delle vulnerabilità (vulnerability disclosure policy).

- Un registro dei sub-processori (chi tratta i dati per conto di TrustedPAI), log di audit, un piano formale di risposta agli incidenti e di notifica delle violazioni dati.
- Per clienti finanziari regolamentati nell'Unione Europea: conformità DORA (obbligatoria dal 2025, spesso richiesta per contratto).
- Contratti standard MSA/SLA, un'assicurazione E&O/cyber liability, un marchio registrato.
- Una vera SLA di uptime con penali contrattuali, una status page pubblica, l'eliminazione del singolo punto di guasto sul database, un piano di disaster recovery testato per davvero, un processo di reperibilità (on-call).
- Un ambiente sandbox separato per i clienti enterprise, webhook asincroni, supporto SSO/SAML per il login aziendale.
- Questionari di sicurezza standard del settore pre-compilati (CAIQ/SIG), case study pubblici di clienti reali.

14.7 Idee interessanti ma rimandate deliberatamente

- Conferma in tempo reale all'utente finale, sopra una certa soglia di rischio o importo.
- Spiegazioni delle decisioni in linguaggio naturale (rilevante anche per la normativa europea sull'intelligenza artificiale, l'AI Act).
- Una vera rete di reputazione condivisa tra clienti diversi, con logica propria: oggi lo storico è condiviso “di fatto” per lo stesso agente, ma senza una rete/prodotto a sé.
- Correlazione delle anomalie TRA agenti/negozi diversi (non solo lo storico del singolo agente), per scoprire pattern di frode coordinata, un'idea confermata anche da una ricerca indipendente sulla sicurezza di AP2, letta durante lo sviluppo.
- Una garanzia assicurata sulla decisione di rischio.
- Un SDK/plugin dedicato per i principali framework di costruzione di agenti AI.

15. L'obiettivo dietro il progetto

TrustedPAI non è pensato per generare semplicemente una piccola rendita mensile stabile. L'obiettivo dichiarato è costruire un'azienda solida e “appetibile”, tale da poter essere eventualmente acquisita in futuro da un player più grande del settore dei pagamenti: una banca, un circuito carte, una società di gestione del rischio pagamenti.

Per questo motivo tutto, il codice, la documentazione tecnica, i nomi dei campi nell'API, persino l'identità pubblica del progetto (username, autore dei commit), è stato scritto e organizzato pensando fin da subito a un pubblico e a un mercato internazionali, non solo italiani: un dettaglio in italiano qua e là sarebbe stato un ostacolo concreto in una futura due diligence da parte di un acquirente non italiano.

Il percorso di ingresso sul mercato discusso, ma non ancora avviato, prevede: prima un pilota in modalità “shadow” con 1-2 clienti trovati tra chi già lavora con ACP/AP2; poi un modello a consumo per API; poi eventuali integrazioni con PSP o agenzie che già gestiscono negozi Shopify come canale di distribuzione; infine, più avanti, eventuali garanzie assicurative costruite sopra lo storico di rischio accumulato nel tempo. È deliberato che nessuno di questi passi sia stato ancora avviato: la decisione presa è di completare prima le funzionalità dell'“arsenale” (sezione 10) e la copertura dei protocolli (sezione 5), per non presentarsi a un primo cliente con un prodotto a metà.

Chi arriva alla fine di questo documento dovrebbe avere, a questo punto, tutti gli elementi per capire non solo COSA fa TrustedPAI oggi, ma anche PERCHÉ è costruito così, dove sono i suoi confini reali, e in che direzione è pensato per crescere.

Appendice: Glossario completo, in ordine alfabetico

Tutti i termini tecnici usati nel documento, raccolti qui per una consultazione rapida.

- **Agente AI**: un programma basato su intelligenza artificiale capace di eseguire azioni autonome (navigare siti, compilare moduli, completare acquisti) per raggiungere un obiettivo assegnato da una persona, senza bisogno di conferma passo-passo.
- **API**: “Application Programming Interface”: un modo standard con cui due programmi si parlano tra loro via internet, mandandosi messaggi in un formato concordato (di solito JSON).
- **Blockchain**: un registro condiviso e pubblico di transazioni, mantenuto insieme da tanti computer diversi, dove ogni operazione registrata è praticamente impossibile da cancellare o alterare in un secondo momento.
- **Chargeback**: la richiesta, fatta dal titolare di una carta alla propria banca, di annullare un pagamento già effettuato e riottenere i soldi indietro, tipicamente per una contestazione o una frode.
- **Chiave pubblica / chiave privata**: una coppia di codici collegati matematicamente: la chiave privata resta segreta e serve per firmare, la chiave pubblica si può condividere con chiunque e serve per verificare che una firma sia autentica.
- **Commercio agentico**: la fetta di commercio online in cui l'acquisto è avviato o completato da un agente AI per conto di una persona, invece che da un click umano diretto in quel preciso momento.
- **Credenziale digitale verificabile**: un documento digitale firmato crittograficamente che attesta un fatto (es. “questo pagamento è autorizzato da questo utente”) e che chiunque può verificare matematicamente senza dover contattare chi lo ha emesso.
- **ECDSA**: un algoritmo di firma crittografica basato sulla matematica delle curve ellittiche, alla base di molte firme digitali moderne, incluse quelle usate dai wallet blockchain.
- **EIP-712**: uno standard Ethereum per firmare messaggi strutturati e leggibili (non solo blocchi di dati grezzi), usato sia da x402 sia da MPP/“tempo”.

- **Endpoint**: un indirizzo specifico di un'API a cui un programma può mandare una richiesta per ottenere un certo servizio.
- **Fail-safe**: un principio di progettazione per cui, in caso di errore o dubbio, il sistema sceglie sempre l'opzione più prudente (es. bloccare o trattenere), mai quella più permissiva.
- **Firma crittografica / firma digitale**: un codice matematico generato con una chiave segreta, allegato a un messaggio, che dimostra chi ha creato quel messaggio e che nessuno l'ha modificato dopo.
- **Hash (impronta digitale)**: un codice di lunghezza fissa calcolato a partire da un contenuto qualsiasi: anche il minimo cambiamento nel contenuto originale produce un hash completamente diverso, il che lo rende utile per scoprire manomissioni.
- **HTTP**: il protocollo standard con cui i browser e le applicazioni comunicano con i server web, alla base di internet come lo conosciamo.
- **JWT (JSON Web Token)**: un formato standard e molto diffuso per rappresentare informazioni firmate digitalmente in modo compatto, usato ad esempio per il CartMandate di AP2.
- **JWE (JSON Web Encryption)**: uno standard per cifrare (non solo firmare) un messaggio, in modo che solo chi possiede la chiave giusta possa leggerne il contenuto; usato dalla modalità “card” di MPP.
- **L2 / Layer 2**: una rete blockchain costruita “sopra” una rete principale (tipicamente Ethereum) per essere molto più veloce ed economica, mantenendo comunque un forte legame di sicurezza con la rete principale.
- **Mandato**: nel linguaggio di AP2, un contratto digitale firmato crittograficamente che autorizza una parte specifica del processo di acquisto (intento, carrello, pagamento).
- **MCP (Model Context Protocol)**: uno standard aperto creato da Anthropic che permette a un assistente AI di usare strumenti esterni in modo strutturato, come farebbe con una cassetta degli attrezzi condivisa.
- **Merchant**: il negozio, cioè chi vende il prodotto o servizio e riceve il pagamento.

- **Nonce**: un numero (o codice) usato una sola volta all'interno di un protocollo crittografico, per garantire che ogni messaggio/autorizzazione sia unico e non riutilizzabile.
- **OAuth 2.0**: uno standard molto diffuso per delegare permessi di accesso in modo sicuro e revocabile, senza condividere direttamente le proprie credenziali (es. “accedi con Google”).
- **Onboarding**: il processo con cui un nuovo cliente (un negozio) viene registrato e configurato su un servizio, la prima volta.
- **PSP (Payment Service Provider)**: l'azienda che gestisce tecnicamente i pagamenti per conto di un negozio (es. Stripe): incassa, gestisce le carte, si occupa della parte finanziaria vera e propria.
- **Rate limiting**: un meccanismo che limita quante richieste un singolo utente/chiave può fare in un dato intervallo di tempo, per prevenire abusi e attacchi automatizzati.
- **Replay attack (attacco di riproduzione)**: un attacco in cui qualcuno intercetta un'autorizzazione di pagamento già usata legittimamente una volta, e prova a rimandarla identica una seconda volta per pagare due volte con la stessa firma.
- **Sandbox**: un ambiente di prova separato da quello reale, dove si può testare un'integrazione o un flusso di pagamento senza muovere soldi veri.
- **SD-JWT-VC**: un formato di credenziale digitale verificabile, basato su JWT, pensato per poter rivelare selettivamente solo alcune informazioni contenute al suo interno; usato dal PaymentMandate di AP2.
- **Stablecoin**: una criptovaluta il cui valore è agganciato a una moneta tradizionale (di solito il dollaro USA), pensata per non oscillare come Bitcoin: 1 stablecoin USDC vale sempre circa 1 dollaro.
- **Token (gettone)**: un codice che rappresenta in modo sicuro un diritto o un permesso (es. il diritto di effettuare un pagamento specifico), senza esporre direttamente i dati sensibili sottostanti.
- **TTL (Time To Live)**: il tempo massimo per cui un'autorizzazione o un dato resta valido, scaduto il quale smette automaticamente di funzionare.

- **Wallet**: un “portafoglio digitale”: un indirizzo su una blockchain, controllato da una chiave privata, dove si tengono e si spostano fondi in criptovaluta.
- **Webhook**: un meccanismo con cui un servizio notifica automaticamente un altro servizio quando succede un certo evento (es. “ordine spedito”), senza che quest'ultimo debba continuare a chiedere aggiornamenti.
- **X-API-Key**: il nome dell'header HTTP usato da TrustedPAI per ricevere la chiave di autenticazione assegnata a ciascun cliente.

Indice