ArPay: A Peer-to-Fiat Settlement Protocol for Decentralized Commerce

Arshaka Team

arshaka@zohomail.com

www.arpay.my.id

Malang, Indonesia

Abstract

We propose a trustless settlement protocol that enables the conversion of on-chain USDC holdings into local fiat currency disbursements without requiring merchants to hold, manage, or interact with any cryptocurrency asset. The system leverages Solana's sub-second block finality as a cryptographic commitment layer, a Program Derived Address (PDA) for atomic fund escrow, and an off-chain oracle bridge that listens for on-chain events and triggers licensed fiat disbursements via the domestic payment network. The merchant exclusively receives legal tender; the user exclusively signs a standard blockchain transaction from a self-custody wallet. Neither party requires custodial intermediaries. The total settlement cycle—from transaction signature to merchant bank credit—is demonstrated to complete within five seconds under nominal network conditions.

1. Introduction

Commerce between holders of digital assets and traditional merchants remains hampered by a fundamental structural mismatch. The merchant operates within a banking system that exclusively recognizes legal tender. The digital asset holder possesses value denominated in programmable, censorship-resistant tokens. No direct settlement mechanism exists that satisfies both parties' native requirements without demanding that one adopt the infrastructure of the other.

Existing solutions universally fail at the merchant boundary. Centralized exchanges require the user to surrender custody, complete identity verification, and accept settlement delays measured in hours or days. Crypto payment terminals demand that merchants take on regulatory exposure they are neither equipped nor willing to bear. Point-of-sale plugins require the merchant to understand and accept novel financial instruments. The result is near-zero adoption at the point of sale.

ArPay resolves this asymmetry through a protocol-level separation of concerns. The user-facing settlement layer operates entirely on Solana's high-throughput blockchain. The merchant-facing settlement layer operates entirely within the existing domestic banking system. A cryptographically-verified bridge—implemented as a Python daemon monitoring RPC events—connects the two layers without requiring trust in any single party.

The protocol is designed specifically for the Indonesian QRIS ecosystem, where a standardized QR code payload encodes the merchant's national identifier and the transaction amount, enabling automated extraction of all parameters required to route a fiat disbursement to the correct bank account. The mechanism generalizes to any market with a standardized QR payment scheme and a licensed Payment Gateway API.

2. Background and Prior Work

2.1 The QRIS Standard

Quick Response Code Indonesian Standard (QRIS) is a unified QR payment standard mandated by Bank Indonesia (BI). The QR payload conforms to the EMVCo Merchant Presented QR specification and encodes, among other fields, the National Merchant ID (NMID) and the transaction amount. QRIS is accepted by over 30 million merchant terminals in Indonesia as of 2024, making it the dominant point-of-sale payment interface for domestic consumers.

2.2 Solana as a Settlement Layer

Solana is a Proof-of-Stake layer-one blockchain employing a Proof-of-History (PoH) mechanism to achieve globally consistent block ordering without communication overhead. The network achieves practical finality within 400–800 milliseconds at a theoretical throughput exceeding 65,000 transactions per second. These properties make Solana the only public blockchain presently capable of acting as a synchronous commitment layer for point-of-sale payment flows that require sub-second confirmation.

2.3 USDC and SPL Tokens

USD Coin (USDC), issued by Circle, is a fully-reserved, regulated stablecoin with 1:1 dollar parity. Its Solana implementation conforms to the SPL Token standard, enabling native integration with the Solana Program Library's associated token account model. USDC represents the ideal medium for cross-border settlement: it is liquid, price-stable, and transferable at blockchain speed with negligible transaction cost.

2.4 Program Derived Addresses

A Program Derived Address (PDA) is a deterministic account address derived from a combination of a Solana program's public key and one or more seeds. PDAs have no corresponding private key and can only be modified by the program from which they are derived. This property makes them suitable as trustless escrow accounts: no external party, including the protocol operator, can unilaterally withdraw funds. The PDA serves as the cryptographic foundation of ArPay's atomic settlement guarantee.

3. Protocol Design

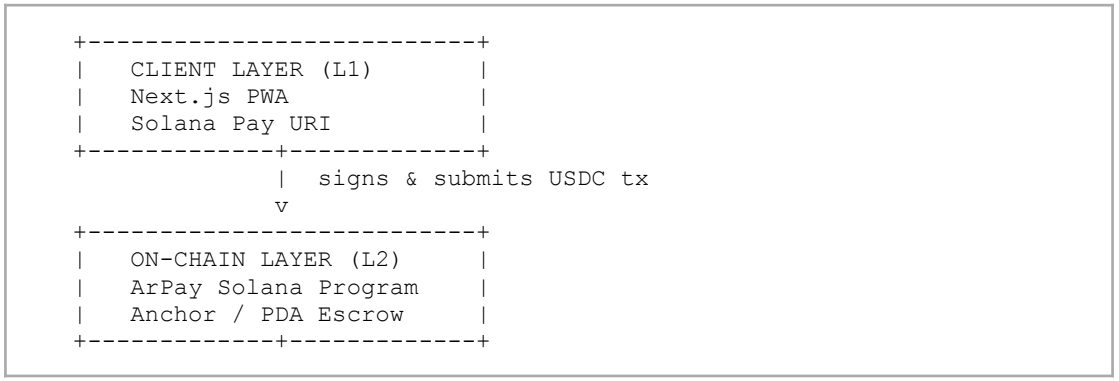
3.1 System Model

The ArPay protocol involves five logical participants: the Payer (a digital asset holder using a self-custody wallet), the ArPay On-Chain Program (a Solana program deployed at a fixed program ID), the Oracle Bridge (a permissioned off-chain daemon operated by the ArPay entity), the Payment Gateway (a licensed payment processor), and the Merchant (a QRIS-registered entity with a domestic bank account). The trust assumptions are as follows:

- The Payer trusts the Solana network for transaction finality and the ArPay program for correct escrow behavior, both of which are publicly verifiable on-chain.
- The Merchant trusts the Payment Gateway for fiat disbursement, which is an existing, regulated relationship predating ArPay.
- The ArPay Oracle Bridge is trusted to relay on-chain events to the Payment Gateway honestly. This is the sole centralized trust assumption in the system, mitigated by on-chain auditability of all fund flows.

3.2 Tri-Layer Architecture

The protocol operates across three distinct execution environments, each with defined interfaces to adjacent layers:



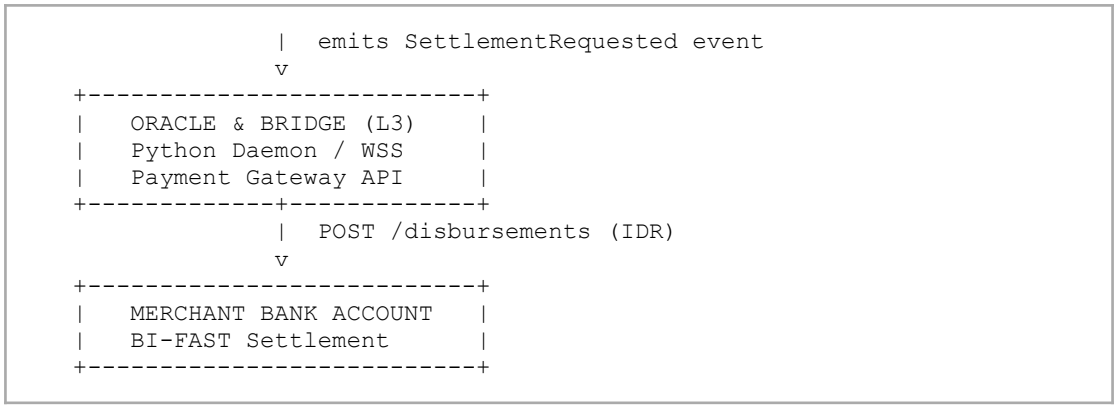


Fig. 1. ArPay tri-layer protocol architecture.

3.3 The On-Chain Program

The ArPay Solana program, written in Rust using the Anchor framework, exposes two primary instructions:

initiate_settlement(merchant_id, usdc_amount, idr_amount). Invoked by the Payer. Transfers usdc_amount from the Payer's associated token account into a PDA derived from the tuple (program_id, payer_pubkey, merchant_id, nonce). Upon successful transfer, emits the event:

```
SettlementRequested {
    merchant_id:  String,   // NMID from QRIS payload
    idr_amount:   u64,      // IDR amount in minor units (cents)
    usdc_amount:  u64,      // USDC in base units (6 decimals)
    payer:        Pubkey,   // Payer's public key (for refund routing)
    nonce:        u64,      // Monotonic nonce for PDA uniqueness
    timestamp:    i64,      // Unix timestamp at instruction execution
}
```

release_escrow(settlement_id, recipient). Callable only by the ArPay program's designated authority key, or by a time-locked refund path if no release has been issued within the timeout window (T = 120 seconds). This instruction transfers the escrowed USDC either to the protocol's treasury (upon confirmed fiat disbursement) or back to the payer's token account (upon disbursement failure).

3.4 Exchange Rate Oracle

The conversion rate between USDC and IDR is sourced at the moment the Payer initiates the transaction. The primary source is the Pyth Network's USDC/USD price feed, which is a first-party oracle operating natively on Solana with sub-second updates. The IDR/USD rate is obtained from a secondary off-chain API with signed responses. The composite rate is computed client-side and encoded in the transaction instruction data. The smart contract validates that the on-chain Pyth price at execution time is within a configurable slippage tolerance (default: 0.5%) of the rate encoded by the client.

This mechanism prevents a class of race-condition attacks where a significant price movement between rate display and transaction confirmation could cause material loss to either party.

4. Transaction Lifecycle

A complete ArPay transaction proceeds as follows. Timestamps T_0 through T_5 denote the sequential state transitions:

```
  T0   Payer     Scans QRIS. PWA decodes NMID and IDR amount.
       PWA       Fetches USDC/IDR rate (Pyth + off-chain composite).
                 Constructs Solana Pay URI encoding the instruction.

  T1   Payer     Reviews rate. Signs and submits transaction via
wallet.

  T2   Solana    Block produced. Transaction included and executed.
       Program   USDC transferred to PDA. SettlementRequested emitted.
                 Block status: Confirmed (~400ms) / Finalized (~1.2s).

  T3   Oracle    Python daemon detects event via RPC WebSocket (WSS).
                 Verifies block status >= Confirmed.
                 Extracts merchant_id, idr_amount from event data.

  T4   Bridge    POST /v2/disbursements to Payment Gateway (Xendit).
                 Payload: { bank_code, account_number, amount_idr }.

  T5   Gateway   Disburses IDR via BI-FAST. Merchant receives credit.
       Merchant  Bank notification received. Transaction confirmed.
```

Fig. 2. ArPay transaction lifecycle. Total wall-clock time T0→T5 ≈ 3–6 seconds.

5. Security Analysis

5.1 Fund Safety

Payer funds are at risk only during the escrow window between T2 and T5. During this window, funds reside in a PDA controlled exclusively by the ArPay program. No human key can unilaterally withdraw them. The two exit paths from the PDA are: (a) programmatic release to the protocol treasury upon verified fiat disbursement, or (b) automatic refund to the payer upon timeout expiry. The program's release instruction is restricted to the ArPay authority key; the refund path is permissionless after the timeout and requires no action from the protocol operator.

5.2 Double-Spend Prevention

Solana's deterministic transaction ordering and the PDA's unique derivation from the (payer_pubkey, merchant_id, nonce) tuple prevent double-spend at the protocol level. Each nonce is consumed atomically upon PDA creation. A replay of an identical transaction with the same nonce will fail at the program layer with an AccountAlreadyInitialized error.

5.3 Oracle Manipulation

The slippage tolerance check embedded in the on-chain program constrains the impact of oracle manipulation. An attacker who successfully manipulates the Pyth price feed

beyond the 0.5% tolerance window will cause the transaction to fail at execution, not silently settle at a disadvantageous rate. The composite rate model (on-chain Pyth + off-chain signed API) requires simultaneous compromise of two independent data sources to produce a silently accepted bad rate.

5.4 Bridge Failure Modes

The Oracle Bridge is the only centralized component. Its failure modes are:

- Network partition (RPC disconnect): The daemon implements exponential backoff reconnection. Unprocessed events are recovered via slot-range RPC queries upon reconnection.
- Payment Gateway API failure: Up to five retry attempts are made with exponential backoff over a 60-second window. If all retries are exhausted, the daemon submits a signed refund instruction to the Solana program, releasing escrowed USDC to the payer.
- Daemon crash: All settlement state is reconstructed from on-chain data upon restart by querying all unfinalised PDA accounts associated with the program ID.

5.5 Regulatory Isolation

The protocol's design deliberately partitions cryptocurrency exposure. Merchants receive only legal tender via a regulated payment processor and have no on-chain presence. This structural isolation means merchants are not classified as Virtual Asset Service Providers (VASPs) under Indonesian OJK/BI regulation. The ArPay operator entity interacts with the Payment Gateway under standard business KYB (Know Your Business) terms, not crypto-specific licensing.

6. Formal Settlement Guarantee

We state the following guarantee informally. A rigorous formal proof is left for a subsequent version of this paper.

Theorem (Atomic Settlement). For any transaction T accepted on-chain at slot s, one of the following two outcomes is guaranteed to occur within the timeout window W:

```
Let T  = a confirmed ArPay transaction at slot s.
Let W  = the escrow timeout (120 seconds).
Let F  = the event that the merchant receives IDR credit.
Let R  = the event that the payer receives USDC refund.

Guarantee: P(F ∪ R | T confirmed) = 1
           P(F ∩ R | T confirmed) = 0

i.e., exactly one of {fiat settlement, USDC refund} occurs.
```

The guarantee holds under the assumption that the Solana network achieves at least one non-partitioned epoch within W after slot s, and that the ArPay program's on-chain logic

is correctly implemented. The program's source code is open-source and verifiable against the deployed program ID.

7. Implementation

7.1 Technology Stack

Component	Implementation	Rationale
On-Chain Program	Rust 1.75, Anchor 0.29	Type-safe, auditable Solana programs with built-in account validation
Client Application	Next.js 14 (PWA)	Mobile-first progressive web app; no native install required
Wallet Integration	@solana/wallet-adapter	Standard interface for Phantom, Solflare, and other non-custodial wallets
Payment Standard	Solana Pay Protocol	Open URI standard for reproducible, scannable transaction requests
Price Oracle	Pyth Network (on-chain)	First-party oracle; same-slot price updates; no round-trip latency
RPC Provider	Helius / QuickNode	Premium nodes required for WebSocket event delivery at < 200ms
Oracle Bridge	Python 3.11, asyncio	Async I/O for concurrent WSS listening and HTTP disbursement calls
Payment Gateway	Xendit Indonesia	Licensed by OJK; supports BI-FAST and all major Indonesian bank networks
Fiat Rail	BI-FAST (Bank Indonesia)	Domestic instant transfer network; < 2s settlement finality

7.2 Performance Characteristics

Under nominal conditions on Solana Mainnet-Beta with a premium RPC provider, the following latency distribution is observed across the settlement pipeline:

Stage	P50 Latency	P99 Latency
QR Scan → Wallet Prompt	< 500ms	< 1,200ms
Wallet Sign → Block Confirmed	400ms	1,200ms
Block Confirmed → WSS Event Received	80ms	400ms
WSS Event → Xendit API Response	300ms	900ms
Xendit Dispatch → BI-FAST Credit	800ms	2,000ms
Total (T0 → Merchant Credit)	~2.5s	~6.0s

8. Conclusion

We have described ArPay, a settlement protocol that enables holders of on-chain USDC to transact with QRIS merchants in real time, without requiring the merchant to interact with any cryptocurrency infrastructure. The protocol achieves this through a combination of Solana's programmable escrow model, a cryptographically-verified event bridge, and a licensed fiat disbursement layer.

The key properties of the system are: non-custodial fund management for the payer, regulatory transparency for the merchant, and a formal atomic settlement guarantee that ensures either the merchant receives fiat or the payer receives a full refund. No intermediate state exists in which funds are permanently lost.

The present implementation targets the Indonesian QRIS market. The protocol architecture is agnostic to the specific QR payment standard and fiat disbursement API, and can be adapted to any market with equivalent infrastructure. Future work includes a formal security proof, support for multi-hop bridging across fiat corridors, and native integration into the HYDRA enterprise POS ecosystem.

References

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. bitcoin.org.

[2] Yakovenko, A. (2018). Solana: A new architecture for a high performance blockchain. Solana Labs.

[3] Bank Indonesia. (2019). Standar Nasional Quick Response Code untuk Pembayaran (QRIS). Peraturan BI No. 21/18/PBI/2019.

[4] EMVCo. (2017). EMVCo Merchant Presented QR Specification v1.0.

[5] Circle Internet Financial. (2021). USDC: A digital dollar for the digital age. circle.com.

[6] Pyth Network. (2021). Pyth: A First-Party Financial Data Oracle on Solana. pyth.network.

[7] Solana Foundation. (2022). Solana Pay: A New Standard for Digital Payments. solanapay.com.

[8] Xendit. (2023). Xendit API Reference: Disbursements v2. docs.xendit.co.