Vektes Protocol

No chargebacks. No sender reversals.

A Payment Protocol with Scheduled Settlements, Transaction Deduplication, and Sender-Side Finality

Whitepaper v1.2 — July 2026

vektes.com

> This document supersedes Whitepaper v1.0. Changes in v1.2: **Section 6 (Fee Model)** now describes oracle-based USD fee tiering, and **Section 3.6** documents recipient rejection — both as implemented in the `VektesProtocol` contract. The project was formerly named "Settl".

Table of Contents

1. Abstract
2. Problem Statement
3. Vektes Protocol
4. Architecture
5. Tokenomics
6. Fee Model
7. Use Cases
8. Security
9. Roadmap
10. Team & Community
11. Conclusion

CRYP2NOVA

---

1. Abstract

Vektes Protocol is a payment protocol built on Ethereum that introduces two fundamental primitives missing from today's financial infrastructure: **user-defined transaction codes** for deduplication and **settlement dates** for time-locked fund release.

In traditional finance, payments suffer from chargebacks, duplicate processing, multi-day settlement windows, and counterparty risk. Existing blockchain solutions address some of these issues but lack the structured finality and scheduling that institutional users require.

Archived copy · cryp2nova.com

Vektes solves this by making every transfer irreversible *by the sender* upon submission — no chargebacks and no sender-side cancellation — while giving senders precise control over when funds become available to recipients. Combined with per-recipient unique transaction codes that prevent duplicate payments at the protocol level, Vektes creates a settlement layer suitable for B2B payments, payroll, cross-border remittances, and any use case where finality and scheduling are critical.

Key Innovations:

- **Transaction Deduplication** — user-provided unique codes prevent double-payments at the protocol level.
- **Scheduled Settlement** — sender-defined release dates lock funds trustlessly until maturity.
- **Sender-Irrevocable** — no chargebacks and no sender-side reversals; once sent, the sender cannot pull funds back.
- **Recipient Rejection** — a recipient may decline an unwanted pending transfer, refunding the sender (useful for compliance or wrong-payment cases).
- **Multi-Asset Support** — transfer USDC, ETH, or any supported ERC-20 with native protocol guarantees.
- **Oracle-Priced Fee Tiers** — free for small transfers, minimal fees at scale (0.005%–0.02%), tiered by true USD-equivalent volume via Chainlink price feeds.
- **Utility Token** — $VEK is the network's intended gas and fee token, with a fixed 1-billion supply.

Vektes is bootstrapped and community-driven, with no venture capital involvement. The protocol is governed by $VEK token holders, who control fee parameters, treasury allocation, and protocol upgrades through on-chain governance.

---

2. Problem Statement

The global payments industry processes over $2 quadrillion annually, yet its core infrastructure remains plagued by inefficiencies that cost businesses billions in fraud, disputes, delays, and operational overhead. These problems persist because legacy systems were designed for an era of manual reconciliation and trusted intermediaries.

2.1 The Chargeback Problem

Chargebacks cost merchants an estimated $117 billion annually (Mastercard, 2023). Beyond direct losses, businesses face chargeback fees ($20–$100 per dispute), operational costs of contesting claims, and the constant uncertainty of whether received funds will be reversed weeks or months after a transaction. This reversibility creates a fundamental trust asymmetry: merchants deliver goods and services without certainty that payment is final. "Friendly fraud" — where legitimate purchases are disputed — accounts for up to 75% of all chargebacks.

Archived copy · cryp2nova.com

2.2 Duplicate Payment Risk

Enterprise Accounts Payable departments process thousands of invoices monthly. Studies show that 0.1%–0.5% of all B2B payments are duplicates — an error rate that translates to millions in overpayments for large organizations. Existing payment rails have no native concept of idempotency: a sender can submit the same payment instruction twice, and the system will happily execute both.

2.3 Settlement Delays in Traditional Finance

Despite decades of technological advancement, major payment systems still operate with multi-day settlement cycles:

Payment Rail	Settlement Time	Finality
ACH (US)	T+1 to T+3	Reversible (60 days)
SWIFT (International)	T+1 to T+5	Reversible
Credit Card	T+2 to T+3	Reversible (120+ days)
Wire Transfer	Same day	Generally final
Vektes Protocol	**User-defined (instant to T+N)**	**Sender-final (no chargebacks)**

Table 1: Settlement time comparison across payment systems.

2.4 Why Existing Crypto Falls Short

Existing blockchain payment solutions (Bitcoin, stablecoin transfers, Solana Pay) solve the finality problem — transactions are generally irreversible once confirmed. However, they lack critical features for enterprise adoption:

- **No native deduplication** — nothing prevents a user from accidentally sending the same payment twice.
- **No scheduled settlement** — funds arrive immediately or not at all; committing funds now for future release requires custom escrow contracts.
- **No standardized reference codes** — reconciling on-chain transactions to off-chain invoices remains manual.
- **No tiered pricing** — gas fees are the same whether you send $10 or $10 million.

Vektes addresses every one of these gaps at the protocol level.

Archived copy · cryp2nova.com

3. Vektes Protocol

The Vektes Protocol extends standard ERC-20 transfers with two parameters: a unique transaction code and an optional settlement date. Together, these create a payment system with native deduplication and scheduled finality.

3.1 Design Principles

- **Sender Irrevocability** — once submitted, a transfer cannot be cancelled, reversed, or redirected by the sender. Recipients get absolute certainty that a sender cannot claw funds back.
- **Sender Control of Timing** — the sender determines when funds become available, mirroring real-world instruments (checks, letters of credit) but with cryptographic enforcement.
- **Protocol-Level Deduplication** — duplicate prevention is enforced by the settlement contract itself, not the application layer.
- **Public Transparency** — all pending transfers are publicly visible on-chain. Any party can verify the existence, amount, and settlement date of a transfer.
- **Multi-Asset Native** — the protocol is asset-agnostic. USDC, ETH, WETH, or any supported ERC-20 can be transferred with identical guarantees.

3.2 Unique Transaction Codes (Deduplication)

Every transfer requires a user-provided transaction code (`bytes32`). This code serves as an idempotency key scoped to the sender→recipient pair:

```
codeKey = keccak256(sender, recipient, txCode)
require(!usedCodes[codeKey], "Duplicate code")
usedCodes[codeKey] = true
```

Scope: per sender→recipient pair. The same code can be reused for different recipients, but never twice for the same pair.

Purpose: maps directly to real-world reference numbers (invoice IDs, PO numbers, payment references). If your ERP system generates unique invoice numbers, those become your transaction codes — the protocol physically prevents paying the same invoice twice.

3.3 Settlement Dates (Time-Locked Release)

Each transfer includes a settlement date (Unix timestamp) that determines when the recipient can claim the funds:

- **Settlement date = 0 or in the past:** instant transfer; funds arrive immediately.
- **Settlement date in the future:** funds are locked in the protocol contract until the specified date. The recipient can see the pending transfer but cannot access funds early.

Archived copy · cryp2nova.com

- **After settlement date:** the recipient calls `claim(sender, txCode)` to receive funds. A `batchClaim()` function enables claiming multiple transfers in a single transaction.

This enables use cases that currently require expensive escrow services or trusted intermediaries: scheduled payroll, deferred supplier payments, milestone-based releases, and contractual payment terms (Net-30, Net-60).

3.4 Sender Irrevocability

The sender has **no way to cancel, reverse, or redirect** a transfer once submitted:

- The sender's funds are immediately debited and locked.
- No function — for the sender, an admin, or governance — can reverse a transfer back to the sender.
- The recipient is assured that committed funds cannot be clawed back by the sender.

This eliminates chargeback risk entirely and gives recipients the same certainty as receiving physical cash. The only way a pending transfer does not complete is if the **recipient** chooses to decline it (refunding the sender) — never the sender. See Section 3.6.

3.5 Multi-Asset Support

The protocol supports any whitelisted ERC-20 token as well as native ETH. The core `send()` function signature:

```
function send(
    address token,          // ERC-20 address (USDC, WETH, etc.)
    address to,             // Recipient
    uint256 amount,         // Transfer amount
    bytes32 txCode,         // Unique transaction code
    uint256 settlementDate  // Release date (0 = instant)
) external;
```

A parallel `sendNative()` function handles ETH transfers with identical deduplication and settlement semantics. Token support is **curated by default** (each token is explicitly whitelisted by governance) and can optionally be opened to any ERC-20. Every supported token must also have a registered USD price feed (see Section 6).

3.6 Recipient Rejection (Decline)

While a sender can never cancel, the **recipient** of a *pending* (scheduled, unclaimed) transfer may decline it via `rejectTransfer(sender, txCode)`. The locked funds are refunded to the **original sender** — never to anyone else — and the transfer can no longer be claimed.

Archived copy · cryp2nova.com

Only the recipient can call this: the sender cannot reject their own transfer, and no third party can. Rejection is available any time before the transfer is claimed (before or after the settlement date). This is **not a chargeback** — the recipient never receives funds and then reverses them; they simply decline delivery, returning value to the sender. It supports real needs such as compliance/sanctions holds, KYC screening, or returning a mistaken payment without an off-chain negotiation. Rejection is enforced at the contract level with a dedicated `cancelled` flag that `claim`, `batchClaim`, and `isClaimable` all honor.

---

4. Architecture

Vektes Protocol is deployed as a set of interconnected smart contracts. The network roadmap (Section 9) targets an Ethereum Layer 2 built on the OP Stack, with $VEK as the native gas token; the protocol contracts themselves are EVM-equivalent and deploy on any EVM chain.

4.1 Smart Contract Architecture

Contract	Purpose	Key Functions
`VektesProtocol`	Core settlement logic	`send()`, `sendNative()`, `claim()`, `batchClaim()`
`VektesToken`	Gas & governance token	ERC-20 + ERC-2612 Permit + Burn
`VektesVesting`	Token distribution	`createSchedule()`, `release()`, `revoke()`

Table 2: Core smart contracts.

The contracts are built on OpenZeppelin libraries (`Ownable`, `ReentrancyGuard`, `Pausable`, `SafeERC20`) and compile with Solidity 0.8.28 targeting the Cancun EVM.

4.2 Transaction Lifecycle

A typical Vektes transfer follows this lifecycle:

1. **Initiation** — sender calls `send()` with token, recipient, amount, txCode, and settlementDate.
2. **Validation** — protocol verifies: amount > 0, recipient ≠ zero address, token supported, code unused (dedup).
3. **Pricing** — the transfer amount is converted to a USD-equivalent via the token's Chainlink price feed (Section 6).
4. **Fee Calculation** — a tiered fee is computed from the sender's rolling monthly **USD** volume.
5. **Fund Lock** — tokens are transferred from the sender to the protocol contract.
6. **Fee Distribution** — the protocol fee is split: 50% burned, 50% to treasury.
7. **Settlement** — for scheduled transfers, a `Transfer` record is stored on-chain (publicly visible); after `settlementDate`, the recipient calls `claim()`.
8. **Completion** — the transfer is marked claimed and funds are released to the recipient.

Archived copy · cryp2nova.com

For instant transfers (`settlementDate` = 0 or past), steps 7–8 are replaced by immediate delivery to the recipient in the same transaction. Alternatively, a recipient may call `rejectTransfer()` on a pending transfer, refunding the sender and closing it out (see §3.6).

4.3 Network (Roadmap)

The Vektes Network targets an optimistic rollup on the OP Stack, inheriting Ethereum security via fraud proofs and data availability posted to L1, with a sequencer for transaction ordering and $VEK as the native gas token. See Section 9 for phasing.

---

5. Tokenomics

5.1 Token Overview

Parameter	Value
Token Name	Vektes
Ticker Symbol	$VEK
Standard	ERC-20 + ERC-2612 (Permit)
Total Supply	1,000,000,000 (1 Billion)
Supply Model	Fixed — no minting after deployment
Fee Handling	A share of protocol fees is routed to a burn address and the treasury (see §6.3)

5.2 Allocation & Vesting

Category	%	Tokens	Vesting
Community & Ecosystem	40%	400,000,000	10% at TGE, 4-year linear
Treasury / DAO	25%	250,000,000	6-month cliff, 3-year linear
Founding Team	15%	150,000,000	1-year cliff, 3-year linear
Liquidity	10%	100,000,000	Deployed progressively (see note)
Early Supporters	7%	70,000,000	25% at TGE, 1-year linear
Advisors	3%	30,000,000	1-year cliff, 2-year linear

Table 3: Token allocation and vesting schedule.

Archived copy · cryp2nova.com

All allocations are minted once at deployment to their respective addresses; the `VektesToken` contract has no mint function, so supply is permanently fixed at 1 billion. Vesting schedules are enforced on-chain by `VektesVesting`, which supports an initial TGE release, a cliff, and linear vesting thereafter.

Note on the Liquidity allocation. The 10% liquidity allocation is *earmarked* for market liquidity and is **deployed progressively**, not all at once. An initial tranche seeds the launch pool (paired against stable-side capital); the remainder is held transparently in the project multisig and deployed over time to deepen liquidity and support additional venues as the protocol grows. Because on-chain liquidity requires matching both sides of a pair, the pace of deployment is bounded by available paired capital — adding token-only liquidity would create a one-sided sell wall rather than genuine depth. Deployed liquidity positions are subject to time-locks (the initial pool is locked for 12 months).

5.3 Utility (Gas, Fees, Governance)

- **Protocol Fees:** $VEK is the settlement fee token — every paid-tier transfer charges its protocol fee in $VEK, pulled from the sender and split between a burn address and the treasury.
- **Gas Token:** $VEK is also the intended native gas token on the Vektes Network; network transactions consume $VEK for execution fees.
- **Governance:** token holders participate in setting protocol parameters — fee tier thresholds and rates, treasury spend, the supported-token whitelist and price feeds, and protocol upgrades.

5.4 Protocol Fee Handling

Protocol fees collected on settlements are split at the contract level: a configurable share is sent to a burn address (`0x…dEaD`) and the remainder to the treasury (currently 50/50; a governance parameter — see §6.3). Supply is fixed at 1 billion with no minting. This is an operational fee-routing mechanism and is not a representation about the token's price.

---

6. Fee Model

Vektes employs a tiered fee structure that makes small transfers completely free while capturing value from high-volume institutional users. **Tiers are denominated in US dollars and enforced on-chain using price oracles**, so a sender's tier reflects true economic volume regardless of which token or how many decimals it uses.

6.1 Tiered Fee Structure

Protocol fees are selected per sender based on their rolling monthly **USD-equivalent** transfer volume:

CRYP2NOVA

Archived copy · cryp2nova.com

Monthly Volume (per sender)	Protocol Fee	Fee on $50K Transfer
$0 – $10,000	0% (Free)	$0
$10,000 – $100,000	0.005%	$2.50
$100,000 – $1,000,000	0.01%	$5.00
$1,000,000+	0.02%	$10.00

Table 4: Tiered protocol fee schedule.

The tier selects a rate applied to the transfer's USD-equivalent value. **The resulting fee is charged in $VEK** (converted from USD via the $VEK price feed) and pulled from the sender separately — so the **recipient always receives the full transfer amount**, and $VEK is consumed on every paid-tier transfer. Free-tier transfers require no $VEK. Senders can preview the exact $VEK fee on-chain and pass a **maximum-fee cap** with the transfer, so a moved or manipulated price feed can never pull more $VEK than they approved.

6.2 Oracle-Based USD Conversion

Each supported token (and native ETH) has a registered Chainlink USD price feed. On every transfer, the protocol converts the token amount to a 6-decimal USD-equivalent that is accumulated into the sender's monthly volume and used to select the fee tier:

```
usd6 = amount × price × 10^6 / (10^tokenDecimals × 10^feedDecimals)
```

where `price` and `feedDecimals` come from the token's price feed and `tokenDecimals` is the token's own decimal precision (18 for native ETH). The conversion uses full 512-bit intermediate precision (`Math.mulDiv`) to prevent overflow.

Oracle safety. Every price read is validated before use:

- `PriceFeedNotSet` — a token with no registered feed cannot be transferred (fail-closed).
- `InvalidPrice` — a non-positive price answer is rejected.
- `StalePrice` — a price whose last update is older than `stalePriceThreshold` (default 24 hours, governance-adjustable) is rejected.

This design corrects a limitation of earlier drafts, in which tiers were nominally denominated in USD but measured against raw token amounts of differing decimals. With per-token price feeds, tier selection is now economically accurate across all supported assets.

6.3 Fee Distribution (Burn + Treasury)

Protocol fees collected at settlement are split between a burn address and the treasury. The split is a governance parameter (currently 50/50).

Archived copy · cryp2nova.com

Fee Component	Burn %	Treasury %
Protocol Fees	50%	50%

Table 5: Protocol-fee distribution (contract-enforced; split is adjustable by governance).

6.4 Comparison to Traditional Payment Systems

Payment System	Fee on $50K	Settlement	Reversible?
Vektes Protocol	**$2.50 – $10**	**User-defined**	**No (irrevocable)**
Wire Transfer	$25 – $50	Same day	Generally no
ACH	$0.20 – $1.50	1–3 days	Yes (60 days)
Credit Card	$1,450 (2.9%)	2–3 days	Yes (120+ days)
SWIFT International	$35 – $65 + FX	1–5 days	Yes
PayPal / Stripe	$1,450 (2.9%)	1–2 days	Yes (180 days)

Table 6: Fee comparison across payment systems.

---

7. Use Cases

Vektes's combination of deduplication, scheduled settlement, and irrevocability enables real-world financial applications that currently require expensive intermediaries or manual processes.

7.1 B2B Invoice Payments

Businesses paying suppliers can use invoice numbers as transaction codes, making it physically impossible to pay the same invoice twice. Settlement dates can be set to Net-30 or Net-60 terms, locking the payment commitment on day one while releasing funds on the agreed date.

7.2 Scheduled Payroll

Employers can batch-submit an entire payroll run with each payment coded to the pay period (e.g., `EMP-1234-2026-07`) and a settlement date set to payday. Funds are committed immediately (cannot be pulled back), but employees receive funds exactly on schedule.

CRYP2NOVA

Archived copy · cryp2nova.com

7.3 Cross-Border Settlement

International payments via SWIFT take 1–5 days and cost $35–65 per transaction plus FX spreads. Vektes enables instant or scheduled settlement in USDC/USDT at 0.005%–0.02% cost, with irrevocable finality and a clear audit trail via transaction codes.

7.4 Escrow Replacement

Traditional escrow services charge 1–2% and require trusted third parties. Vektes's time-locked transfers serve as trustless escrow: the sender commits funds (irrevocable) with a future settlement date tied to a delivery milestone — no third party needed.

7.5 Subscription & Recurring Payments

Users can pre-authorize a series of future payments by submitting multiple transfers with sequential settlement dates (e.g., the 1st of each month for 12 months). Each uses a unique code (`SUB-SERVICE-2026-01` through `SUB-SERVICE-2026-12`), preventing duplicates while guaranteeing the provider will receive payment on schedule.

---

8. Security

8.1 Smart Contract Security

The Vektes Protocol contracts are designed with defense-in-depth:

- **ReentrancyGuard** — all state-changing external functions are protected against reentrancy.
- **SafeERC20** — handles non-standard ERC-20 implementations safely.
- **Pausable** — an emergency circuit breaker allows the protocol to halt if a vulnerability is discovered.
- **Custom Errors** — gas-efficient, descriptive error handling.
- **Access Control** — an `Ownable` pattern for admin functions, with a planned transition to governance.
- **Input Validation** — all functions validate zero amounts, zero addresses, and token support.
- **Oracle Guards** — price reads are validated for existence, positivity, and freshness (`PriceFeedNotSet` , `InvalidPrice` , `StalePrice`).

8.2 Sender No-Cancellation Guarantee

The protocol's most important security property is what the **sender** cannot do: there is no function, admin override, or governance mechanism that lets a sender (or the owner) cancel, reverse, or redirect a submitted transfer back to the sender. The only party that can stop a pending transfer is the **recipient**,

Archived copy · cryp2nova.com

by rejecting it — in which case the funds are refunded to the original sender, never redirected elsewhere (see §3.6). This preserves chargeback-free finality from the sender's side while giving recipients a compliance-friendly opt-out.

8.3 Curated Token Support

Token support is curated by default. Each supported token must be explicitly whitelisted and must have a registered price feed, which mitigates exposure to malicious or non-standard tokens (including fee-on-transfer tokens). An optional open-support mode exists but is disabled by default and intended only for controlled environments.

8.4 Audit & Formal Verification Strategy

- **Phase 1:** internal review and automated tooling (Slither, Mythril, Echidna fuzzing); comprehensive unit-test suite with high coverage.
- **Phase 2:** professional audit by a top-tier firm (e.g., Trail of Bits, OpenZeppelin, CertiK).
- **Phase 3:** public bug bounty program (Immunefi).
- **Phase 4:** formal verification of critical invariants — no duplicate codes, settlement-date enforcement, fee and oracle-conversion correctness.

---

9. Roadmap

Phase 1: Testnet & Smart Contract Audit (Q3–Q4 2026)

- Deploy `VektesProtocol`, `VektesToken`, and `VektesVesting` to testnet.
- Public testnet with an incentivized testing program.
- Professional smart contract audit (2 independent firms).
- Bug bounty program launch on Immunefi.
- SDK and developer documentation; wallet integration partnerships.

Phase 2: Mainnet Launch & Token Generation (Q1 2027)

- $VEK token deployment on Ethereum L1.
- Vektes L2 mainnet launch (OP Stack with $VEK as gas token).
- Liquidity bootstrapping event (fair launch via LBP).
- Bridge deployment (Ethereum L1 ↔ Vektes L2).
- Initial supported assets: USDC, USDT, WETH, DAI — each with a registered Chainlink feed.
- Block explorer and transaction tracking UI.

Phase 3: Ecosystem Growth & Integrations (Q2–Q4 2027)

- Payment API and merchant SDK.

Archived copy · cryp2nova.com

- ERP integrations (QuickBooks, Xero, SAP).
- Payroll platform partnerships; cross-border remittance corridors.
- Mobile wallet application; developer tooling and documentation.

Phase 4: Decentralization & Governance (2028+)

- Transition from a single sequencer to shared/decentralized sequencing.
- On-chain governance deployment (proposals + voting).
- Progressive decentralization of sequencing; protocol ownership transferred to a DAO.
- Cross-chain expansion and a ZK-proof upgrade path exploration for faster finality.

---

10. Team & Community

Vektes is a bootstrapped, community-first project with no venture capital and no private token sale. Insider allocations (team and advisors) are minority stakes released only through long-cliff, multi-year on-chain vesting. The project's legitimacy comes from its open-source code, transparent tokenomics, and community governance.

Community-First Philosophy. The 40% community allocation — the largest single category — prioritizes broad distribution to users and contributors. Community tokens are allocated for contributions such as testnet participation, development, liquidity provision, and documentation.

Governance Model. Protocol governance is controlled by $VEK token holders through on-chain proposals and voting. Scope includes fee tier thresholds and rates, burn percentage, treasury spend, the supported-token whitelist and their price feeds, and protocol upgrades. A minimum quorum of 4% of circulating supply is required for proposals to pass; a 67% supermajority is needed for protocol upgrades, and a simple majority (50%+1) for parameter changes.

CRYP2NOVA

---

11. Conclusion

The global payment system is broken. Chargebacks cost merchants $117 billion annually. Duplicate payments drain enterprise accounts. Settlement delays tie up working capital for days. And existing blockchain solutions, while offering finality, lack the structured mechanisms that real-world commerce requires.

Vektes Protocol solves these problems at the protocol level with three simple innovations:

- **Unique transaction codes** that make duplicate payments physically impossible.
- **Settlement dates** that give senders precise control over fund release timing.
- **Sender-irrevocable commitment** that eliminates chargeback risk entirely — with a recipient-only rejection path for declining unwanted funds.

Archived copy · cryp2nova.com

Powered by the $VEK utility token, priced by transparent on-chain oracles, and governed by its community — Vektes is the settlement layer the internet has been waiting for.

No chargebacks. No sender reversals.

Website: vektes.com — Contact: team@vektes.com

Archived copy · cryp2nova.com