Ceda

WHITEPAPER · VERSION 1.0

Ceda Whitepaper 1.0

Ceda is your money.

Web ceda.money **Published** November 2026 **Ticker** CEDA · wCEDA

0.2 s	$0	1B	65 MB
to final, measured	fee to send or receive	CEDA, fixed forever	memory for a Node

CONTENTS

1. Abstract
2. Why Ceda
3. Design overview
4. The ledger: one chain per account
5. Keys and addresses
6. Consensus: weighted agreement
7. Proof-of-work against spam
8. The Node
9. The wallet
10. wCEDA and cashing out
11. Free Ceda and Node rewards
12. Ceda and Bitcoin
13. Sources

0 Abstract

Ceda is digital money for everyday payments: free to send, final in a fraction of a second, and usable by anyone with a phone. It is built for the 1.3 billion adults who still have no bank or mobile-money account.

Ceda runs on its own network. It is a fork of Nano's block-lattice design, rebuilt from RsNano (the Rust implementation of the Nano node) and cut down to what a payment network needs. Each account keeps its own chain of signed blocks; Nodes agree on each payment by votes weighted by the Ceda that wallets have chosen to put behind them, and a payment is final once two-thirds of the online weight agrees. There are no miners and no fees: spam is held back by a small proof-of-work on each block, light enough that a phone computes it in under a second.

The supply is fixed at 1,000,000,000 CEDA with 8 decimal places, all created in the first block. A token twin, wCEDA, lives on BNB Smart Chain so that Ceda can be exchanged for digital dollars. This paper describes the design, the data formats and the software, with code from the actual implementation.

1 Why Ceda

The World Bank's Global Findex 2025 counts about **1.3 billion adults** without an account at a bank or a mobile-money provider. Most of them are not without money; a bank is too far away, too expensive, or asks for papers they do not have. The same report holds the opening: about **900 million** of those adults already own a mobile phone, and **530 million** a smartphone.

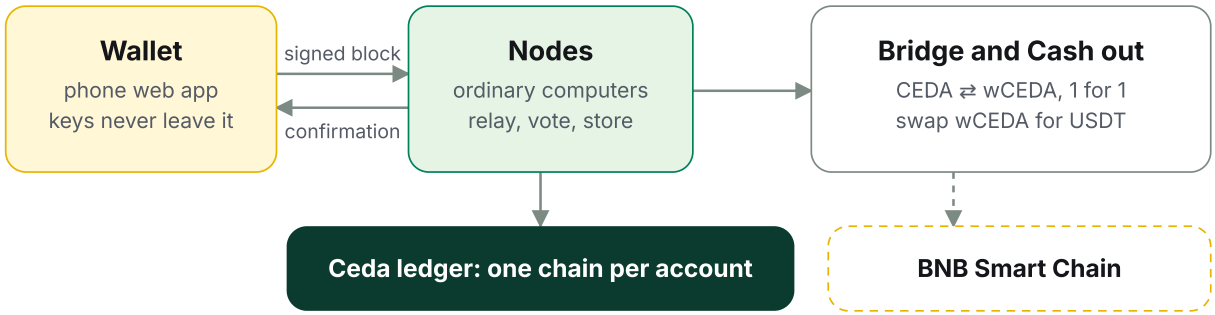
Moving money is also most expensive for those who can least afford it:

Moving $20	What it costs	Source
Sending it to family abroad	about $1.27 (6.36% average)	World Bank, Remittance Prices Worldwide, Aug 2025
A shop taking it by card	$0.88 (2.9% + 30¢)	Stripe standard US rate
Sending it with Ceda	$0	no fee to send or receive

Ceda's goal is money that works like a text message: for anyone with a phone, at no cost, with no account to open.

2 Design overview

Ceda has four parts. Each is described in its own section.



A wallet signs a block and hands it to a Node; Nodes vote, and the block becomes part of the shared ledger. wCEDA on BNB Smart Chain is used only for exchange into other currencies.

- **Ledger.** A block-lattice: every account is its own chain, so payments from different people never wait in one shared queue.
- **Consensus.** Open representative voting: Nodes vote with the weight wallets delegate to them; two-thirds of online weight makes a payment final.
- **Node.** A trimmed RsNano program plus a desktop app with one switch.
- **Wallet.** A web app with no build step, installable without an app store; it signs everything on the device.

3 The ledger: one chain per account

Ceda keeps Nano's block-lattice. Each account has its own chain of **state blocks**, and only the holder of the account's key can add to it. A payment is two blocks: a *send* on the sender's chain, which lowers its balance, and a *receive* on the receiver's chain, which raises theirs. Every block carries the account's whole new balance, so any Node can check a block on its own.

The state block

Field	Size	Meaning
`account`	32 bytes	the account's public key (shown as a `ceda_` address)
`previous`	32 bytes	hash of the account's previous block, zero for its first
`representative`	32 bytes	the Node this account's balance backs (see consensus)
`balance`	16 bytes	the account's balance after this block, big-endian, in raw units
`link`	32 bytes	for a send: the receiver's key; for a receive: the hash of the send
`signature`	64 bytes	Ed25519 with Blake2b-512 in place of SHA-512, over the block hash
`work`	8 bytes	proof-of-work nonce (section 6)

Amounts are whole numbers of **raw**: 1 CEDA = 10^8 raw, and the whole supply is 10^{17} raw, which fits easily in the 128-bit balance field.

TYPES/SRC/CURRENCY_CONSTANTS/CEDA.RS (NODE)

```
pub const ACCOUNT_PREFIX: &str = "ceda";
pub const RAW_PER_COIN: u128 = 10u128.pow(8);
pub const MAX_SUPPLY_RAW: u128 = 1_000_000_000 * RAW_PER_COIN;
```

The block hash is Blake2b-256 over a fixed 32-byte preamble (the block type, 6, in the last byte) followed by the fields in order. This is the wallet's own code:

WALLET/JS/NANO.JS

```
const PREAMBLE = (() => { const p = new Uint8Array(32); p[31] = 6; return p; })();

export function stateBlockHash({ account, previous, representative, balance, link }) {
  return blake2b256(concat(
    PREAMBLE,
    addressToPublicKey(account),
    hexToBytes(previous || ZERO32),
    addressToPublicKey(representative),
    balanceToBytes(balance),      // 16 bytes, big-endian
    hexToBytes(link),
  ));
}
```

A signed send, as the wallet hands it to a Node's `process` command:

A SEND BLOCK (JSON)

```
{
  "type": "state",
  "account": "ceda_3dp5c…j4cd7p",
  "previous": "991CF190094C00F0B68E2E5F75F6BEE95A2E0BD93CEAA4A6734DB9F19B728948",
  "representative": "ceda_1x8izb…ta78qds",
  "balance": "100425000000",          // 1,004.25 CEDA left after the send
  "link": "6A4B…E2C1",                  // the receiver's public key
  "signature": "5B11B17D…E61600",
  "work": "3c82cc724905ee95"
}
```

The amount sent is never written down: it is the difference between this balance and the previous block's. The receiver's wallet (or their Node, for its own wallet) then publishes a receive block whose `link` is the send's hash.

4 Keys and addresses

A wallet starts from 12 recovery words, the 128-bit BIP-39 form. Those 16 bytes are stretched to a 32-byte seed with a keyed Blake2b, and account keys are derived from the seed by index. Wallets from Nano's 24-word form (32 bytes) use the bytes directly as the seed.

WALLET/JS/NANO.JS

```
export function seedFromSecret(secret) {
  if (secret.length === 32) return secret;                    // 24 words
  return blake2b(secret, { dkLen: 32,                        // 12 words
    key: new TextEncoder().encode('Ceda wallet seed v1') });
}

export function deriveSecretKey(seed, index = 0) {
  const idx = new Uint8Array(4);
  new DataView(idx.buffer).setUint32(0, index, false);  // big-endian index
  return blake2b256(concat(seed, idx));
}
```

An address is `ceda_`, then the 256-bit public key in a 32-letter alphabet without look-alike characters (52 letters, padded to 260 bits), then a 40-bit Blake2b checksum (8 letters). Every address is 65 characters long and a typo is caught before anything is sent. Nano addresses (`nano_`, `xrb_`) are refused by name, so money cannot be sent to the wrong network by mistake.

AN ADDRESS, PART BY PART

```
ceda_  3dp5c…(52 letters: the public key)…  j4cd7p…(8 letters: checksum)
// checksum = blake2b(publicKey, { dkLen: 5 }), bytes reversed
```

The secret never leaves the phone. With a passcode it is stored encrypted with AES-GCM under a key from PBKDF2-SHA-256.

5 Consensus: weighted agreement

Because each account writes only to its own chain, the only conflict that can arise is a **fork**: two different blocks claiming the same `previous`, that is, an attempt to spend the same Ceda twice. Nodes do not need to agree on an order for everything; they need to agree, per block, on whether it is the one that stands.

Delegated weight

Every account names a `representative`. The representative's **weight** is the sum of the balances that name it. Delegation moves no money and can be changed by the account at any time with its next block. The wallet delegates to a Node that is online automatically, so everyday users help secure the network without doing anything.

Votes and finality

1. A Node that receives a new block checks its signature, its balance arithmetic and its proof-of-work, then passes it on.

2. Representatives with enough weight sign a **vote** for the block's hash and broadcast it.
3. Each Node tallies the votes by weight. When the votes for one block reach the **quorum, 67% of the online weight**, the block is **confirmed** and written as final (Nano calls this "cementing"). It can never be rolled back.
4. If two blocks fork, the same weighted vote picks one; the other is dropped everywhere.

Ceda changes one timing: a Node re-sends its vote on a block that is still waiting every **5 seconds instead of Nano's 15**, so a lost message costs seconds, not a quarter of a minute.

NODE/CONSENSUS/VOTE_GENERATION/AEC_VOTER.RS

```
// Vote again every 5 s (was 15 s) on payments still waiting:
// a missed vote costs seconds, not 15.
_ ⇒ Duration::from_secs(5),
```

In our measurements two Ceda Nodes confirmed a payment in **0.15 to 0.2 seconds**, and 200 payments sent at once from 200 accounts were all final in **0.48 seconds**.

> Security comes from weight, not from the number of machines: running a thousand Nodes gives no say without Ceda delegated to them. An attacker would need the backing of a third of the online Ceda to stall the network, and two-thirds to confirm a double spend.

6 Proof-of-work against spam

With no fees, something must make flooding the network costly. Each block carries a nonce such that a Blake2b-64 hash of the nonce and the block's **root** (its `previous`, or the account key for a first block) reaches a threshold:

WALLET/JS/NANO.JS

```
export function workValue(rootHex, workHex) {
  const w = hexToBytes(workHex).reverse();     // sent big-endian, hashed little-endian
  const h = blake2b(concat(w, hexToBytes(rootHex)), { dkLen: 8 });
  let v = 0n;
  for (let i = 7; i ⩾ 0; i--) v = (v << 8n) | BigInt(h[i]);
  return v;                                    // valid if v ⩾ threshold
}
```

The expected number of hashes is 2^{64} / (2^{64} − threshold). Nano asks for about 537 million hashes (2^{29}) per send, which takes many seconds on every core of a computer and is out of reach of a phone. Ceda sets its thresholds so that a phone can do the work itself in under a second:

TYPES/SRC/CURRENCY_CONSTANTS/CEDA.RS

```
pub const WORK_THRESHOLD_EPOCH1:         u64 = 0xfffff00000000000; // 2^20 hashes
pub const WORK_THRESHOLD_EPOCH2:         u64 = 0xfffff00000000000; // 2^20, payments sent
pub const WORK_THRESHOLD_EPOCH2_RECEIVE: u64 = 0xffffc00000000000; // 2^18, payments received
```

Block	Ceda	Nano
Payment sent	≈ 1 million hashes (2^{20})	≈ 537 million (2^{29})
Payment received	≈ 262 thousand (2^{18})	≈ 8 million (2^{23})

Lower work does not leave the network open: Nodes also limit traffic per peer and order waiting blocks by the balance and age of their accounts, so a spammer with little Ceda only competes with itself. Lowering a threshold keeps every older block valid.

7 The Node

The Ceda Node is RsNano, the Rust implementation of the Nano node, at its July 2026 code, with everything a payment network does not need removed. Rust rules out whole classes of memory errors, which matters for software that holds money. The consensus rules, account chains and block format were kept unchanged on purpose: that is the one place a trim could break the network, for little gain.

What we removed

	RsNano	Ceda Node
Lines of Rust	131,438	105,331
Program size (release)	11.8 MB	7.8 MB
Built-in packages	337	289
Commands a Node answers	111	22

Gone are 89 commands, callbacks to outside web servers, remote proof-of-work servers (and the web-client library they needed), graphics-card proof-of-work, an experimental snapshot feature, Nano's developer tools and 54 unused libraries. All remaining tests pass. The 22 commands that remain are exactly what a wallet and the Node app use:

THE NODE'S COMMANDS

```
account_balance  account_create  account_history  account_info  backers
block_count  block_info  blocks_info  confirmation_history  confirmation_quorum
peers  process  receivable  send  stop  telemetry  uptime  version
wallet_add  wallet_contains  wallet_create  work_generate
```

ASKING A NODE FOR A BALANCE

```
$ curl -s localhost:2122 -d '{"action":"account_balance",
    "account":"ceda_1x8izbgt6sdkrnpxnzyoiqen3fxdkuac4jx5h3dqgehr5gthwsmg5ta78qds"}'
{ "balance": "1004250000000", "receivable": "0" }
```

Its own network

Ceda Nodes identify themselves with network bytes `'C','C'` (live) and `'C','X'` (test). A Nano Node is turned away at the first handshake. The default ports are 2121 (other Nodes), 2122 (local commands, on 127.0.0.1 only) and 2124 (live updates for wallets).

Light on a personal computer

The Node app runs the Node in the background with small thread pools instead of one thread per processor core, at a lower priority than the person's own apps. A Node used about **65 MB** of memory in our tests, so an ordinary laptop is enough.

CONFIG-NODE.TOML, AS THE NODE APP WRITES IT

```
[node]
io_threads = 2
network_threads = 2
background_threads = 2
work_threads = 1
signature_checker_threads = 1

[node.block_processor]
threads = 1

[node.vote_processor]
threads = 1
```

8 The wallet

The wallet is a web app: it opens from a link and installs to the home screen of an Android phone, an iPhone or a computer in one tap, with no app store, no account and no forms. It is 46 plain files with no

build step and no tracking, and it checks at every start that it is running exactly the files that were published, by comparing a fingerprint of all of them.

- **Keys stay on the phone.** Every block is built and signed on the device; a Node only ever sees signed blocks.
- **Talks to any Node.** A wallet needs only a handful of the Node's commands: `account_info`, `account_history`, `receivable`, `process` and `work_generate`, plus the live-update channel for incoming payments.
- **Simple on purpose.** One balance, Send, Receive and Scan, and a green card the moment money arrives.

WALLET/JS/NANO.JS: BUILD AND SIGN A BLOCK

```
export function makeStateBlock({ secretKey, previous, representative, balance, link, work })
{
  const account = publicKeyToAddress(publicKeyOf(secretKey));
  const fields = { account, previous: previous || ZERO32, representative,
                   balance: BigInt(balance), link };
  const hash = stateBlockHash(fields);
  const signature = bytesToHex(sign(hash, secretKey));      // Ed25519 + Blake2b
  return { hash: bytesToHex(hash), block: { type: 'state', ...fields,
           balance: fields.balance.toString(), signature, work } };
}
```

9 wCEDA and cashing out

A shop that takes Ceda needs a way to turn it into ordinary money. Ceda therefore has a twin on BNB Smart Chain, **wCEDA**: the same Ceda in token form, which can be swapped for USDT, a digital dollar.

wCEDA	
Network	BNB Smart Chain (chain 56), BEP-20
Address	`0x1020e7D1cF2A61894b7fb9721f085056CC9fc6Dc`
Supply	1,000,000,000 wCEDA, 8 decimals, made once at creation
Source code	verified on BscScan, exact match

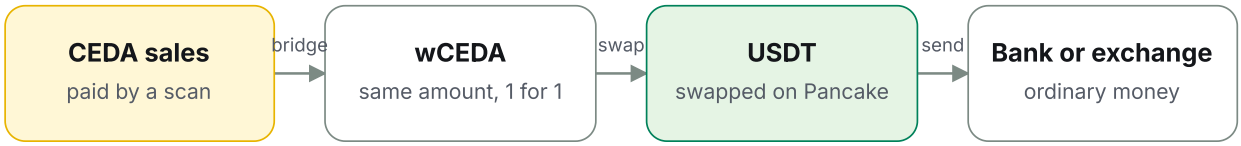
The whole contract is short enough to read in full. It is OpenZeppelin's standard ERC-20 (version 5.6.1), with no owner, no pause, no blocklist and no upgrade:

WRAPPEDCEDA.SOL

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.28;

import {ERC20} from "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract WrappedCeda is ERC20 {
    uint256 public constant TOTAL = 1_000_000_000 * 10 ** 8;

    constructor(address holder) ERC20("Wrapped Ceda", "wCEDA") {
        _mint(holder, TOTAL);
    }

    function decimals() public pure override returns (uint8) {
        return 8;
    }
}
```

One supply, two forms

Nothing is burned and nothing new is made. The bridge holds the twin of every coin that is in use in the other form: when someone turns 100 CEDA into wCEDA, the bridge takes in their 100 CEDA and releases 100 wCEDA from its reserve, and the reverse. So the CEDA and wCEDA in use together never exceed 1 billion. Both use 8 decimals, so a conversion never rounds.



The Node app's Cash out page is planned to do the whole path in one step. The wCEDA/USDT pool is live on Pancake (formerly PancakeSwap): `0x5d1a80820d319c3e66c5ba9da0c462feb314fa07`.

Because Ceda has no memo field, the bridge is designed to give each user their own CEDA deposit address and watch it, so deposits are matched to their wCEDA by address, never by a note. The bridge and the Cash out page are being built and tested now.

10 Free Ceda and Node rewards

Free Ceda for wallets

A new wallet can claim a little Ceda each day with one tap, so a first payment never needs a purchase. The amount grows with each day in a row:

WALLET/TOOLS/FAUCET-RULES.MJS

```
export const DEFAULT_RULES = { first: '0.001', step: '0.001', max: '0.01',
                               everyHours: 24, graceHours: 48 };
// day 1: 0.001 CEDA, day 2: 0.002 … from day 10: 0.01 CEDA a day
```

Node rewards

Every Node that stays on earns a small daily amount, with or without a balance, paid automatically with nothing to claim. While its Node is running and connected to other Nodes, the Node app checks in with the rewards service once an hour:

THE HOURLY CHECK-IN

```
POST /checkin
{ "account": "ceda_1x8izbgt6sdkrnpxnzyoiqen3fxdkuac4jx5h3dqgehr5gthwsmg5ta78qds" }
→ { "ok": true }
```

A Node seen in at least 20 of a day's 24 hours is paid 2 CEDA for that day. Each payment carries an id the Node will not pay twice, so a restart never pays anyone double:

REWARDS/REWARDS-CORE.MJS

```
export function payoutsFor(day, rules = DEFAULT_RULES, exclude = []) {
  const paid = Object.entries(day || {})
    .filter(([account, hours]) => !exclude.includes(account) && hours.length >= rules.minHour
s)
    .map(([account, hours]) => ({ account, hours: hours.length }));
  if (!paid.length) return [];
  const each = toUnits(rules.perNode), budget = toUnits(rules.dailyBudget);
  // past dailyBudget (20,000 CEDA), the day is shared equally
  const amount = each * BigInt(paid.length) > budget ? budget / BigInt(paid.length) : each;
  return paid.map((p) => ({ ...p, amount }));
}

export const payoutId = (key, account) => `ceda-reward-${key}-${account}`;
```

Both come from an ordinary rewards wallet. Its amounts can be adjusted as the network grows.

11 Ceda and Bitcoin

Bitcoin proved that money can work without a bank. It was designed as scarce digital gold, and that design carries costs when it is used to buy bread: fees, waiting and mining. Ceda is designed for the opposite job, everyday payments.

	Bitcoin	Ceda
Fee to send	$0.18 on average on 22 Nov 2026; it rises with demand and neared $60 in 2017	none, at any amount
Time until final	about 10 minutes for one confirmation, about an hour for six	about 0.2 seconds, and it cannot be reversed
How it is kept honest	mining, about 138 TWh of electricity a year	weighted agreement between Nodes; no mining
Who runs it	miners, paid with new coins and fees	ordinary computers running the free Node app
Supply	21 million, issued over time to miners	1 billion, all in the first block
A $1 payment	the average fee alone is 18% of it	free

Payment layers on top of Bitcoin, such as Lightning, are faster and cheaper, but they add channels and balances that users or their providers must manage. Ceda needs none of that: the base network itself is fast and free, so a first-time user with a phone gets the whole benefit with no setup. Ceda does not try to replace Bitcoin as a store of value. It aims to be the money people actually spend.

12 Sources

- World Bank, Global Findex 2025 press release, 16 July 2025
- World Bank, Remittance Prices Worldwide, updated 18 August 2025
- Stripe pricing, standard US card rate
- Nano documentation: protocol design
- RsNano, the Rust Nano Node
- YCharts, Bitcoin average transaction fee, 22 November 2026
- CoinMarketCap, How long does a Bitcoin transaction take?
- Cambridge Judge Business School, Cambridge Digital Mining Industry Report, 28 April 2025
- wCEDA on BscScan (verified source code)

- Ceda measurements: our own tests of the trimmed Node, two Nodes on one two-core Linux computer, November 2026

Ceda is software, not a bank, broker, exchange or adviser, and nothing here is financial advice. Using crypto is risky: its value can change and you can lose what you put in. You are responsible for following the laws where you live.