WHITEPAPER

NumFranc (NFR)

A peer-to-peer digital currency designed from day one to withstand quantum computers

Version 1.1 — October 2026

ShadowRook

Release signing public key (SLH-DSA-SHAKE-128s)

5e1d1c80fdd07d3bc4095f45a05649fc006bce347609b1566b3c25e940881736

Contents

Abstract

1. Why NumFranc

2. Overview

3. Cryptography

4. Monetary issuance

5. Blocks and proof of work

6. Transactions

7. Consensus and chain security

8. Network

9. Wallet

10. Application and distribution

11. Release signing

12. Roadmap

13. Conclusion

References

Appendix A — Consensus and network parameters

Appendix B — Proof-of-work and difficulty computations

Appendix C — Binary transaction format

Appendix D — Text formats

Appendix E — Issuance schedule

Appendix F — Network messages

Appendix G — Glossary

Abstract

NumFranc (symbol **NFR**) is a peer-to-peer digital currency with no bank, no central server and no issuer. It builds on the principles proven by Bitcoin: a public ledger of chained blocks, coins represented as unspent transaction outputs, and a proof of work that makes rewriting the past expensive.

What sets it apart is a choice made **from the very first block**: all of NumFranc's cryptography is post-quantum. Transactions are signed with ML-DSA-87 (NIST FIPS 204, highest security level), every hash uses SHA3-512, and software releases are signed with SLH-DSA (FIPS 205), whose security rests on hash functions alone. A quantum computer able to break Bitcoin's signatures would not be able to steal NFR.

The total supply is fixed at **69,100,000 NFR**, issued over about 98 years at a rate of one block every **691 seconds**. Mining uses Argon2d, a memory-hard function that keeps an ordinary computer competitive. NumFranc also adds features designed for everyday use: readable addresses with a checksum, messages attached to payments, time-locked or cancellable payments, multi-signature, nicknames ("alice.nfr"), and a wallet protected by two passwords and a recovery code split into three pieces.

This document describes NumFranc as implemented in version 1.0 of the software. Features that are only planned are presented separately, in the roadmap.

1. Why NumFranc

1.1 The quantum threat

Bitcoin and most cryptocurrencies sign their transactions with elliptic curves (ECDSA or Schnorr on secp256k1). Their security relies on the hardness of the discrete logarithm. In 1994, Peter Shor showed that a large enough quantum computer would solve this problem in reasonable time. Such a computer could recover a secret key from the public key, then spend the funds in place of their owner.

Machines of the required scale do not exist yet. But a currency must protect funds for decades, and a public key published today in a permanent ledger can be attacked tomorrow. Migrating an existing currency to other signatures is hard: it requires agreement from the whole network, and funds whose owners have disappeared remain exposed.

1.2 The new standards

In August 2024, NIST (the US standards institute) published the first post-quantum cryptography standards, after eight years of public evaluation:

- **FIPS 204 — ML-DSA**: signatures based on lattices (the Module-LWE problem);
- **FIPS 205 — SLH-DSA**: signatures based on hash functions only.

No known quantum algorithm breaks these problems faster than an exhaustive search, which it only slightly speeds up.

1.3 NumFranc's choice

NumFranc has no past to migrate: it is born post-quantum. Its goals, in order of priority, are:

1. **withstand quantum computers**, for every signature and every hash;
2. **withstand 51% attacks** and chain manipulation;
3. **withstand software tampering**, through signed releases;
4. **stay simple to use**: one program, one window, in French or English.

2. Overview

NumFranc is a network of equal nodes. Each node keeps the full blockchain, checks every block and every transaction against the same rules, and relays what it receives. No node has any special authority.

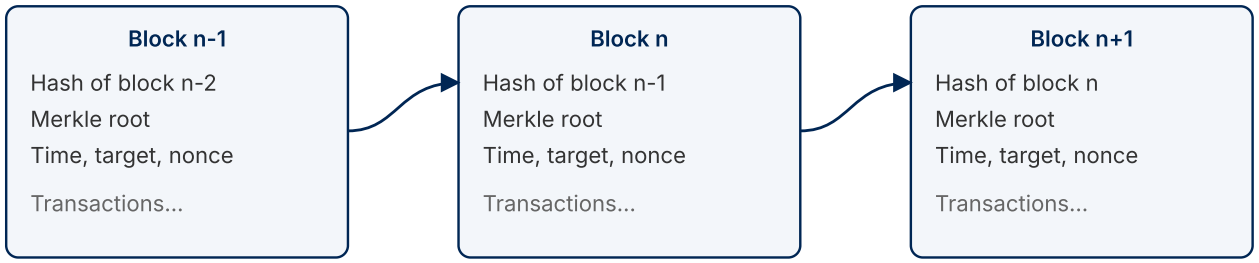


Figure 1. Each header contains the hash of the previous block: changing an old block changes every hash after it.

The main parameters are summarised below. The full list is in Appendix A.

Parameter	Value
Symbol	NFR
Total supply	69,100,000 NFR
Divisibility	1 NFR = 100,000,000 units (8 decimals)
Target time between blocks	691 seconds (11 min 31 s)
Difficulty adjustment	every 691 blocks
Proof of work	Argon2d, 4 MiB of memory per attempt
Transaction signatures	ML-DSA-87 (FIPS 204)
Hashes	SHA3-512 (FIPS 202)
Release signatures	SLH-DSA-SHAKE-128s (FIPS 205)
Maximum block size	8 MiB

Parameter	Value
Network port	6910 (TCP and UDP)
Genesis block	17 October 2026, 00:00:00 UTC

3. Cryptography

NumFranc invents no cryptographic primitive. It combines standardised algorithms, implemented by public, tested Rust libraries. The only computation written for the project is Shamir secret sharing, whose arithmetic is checked by tests.

3.1 Hashes: SHA3-512

Every hash in NumFranc uses **SHA3-512** (FIPS 202), from the Keccak family, whose "sponge" construction differs from SHA-2's. Block and transaction identifiers are **double hashes**, written `H(x) = SHA3-512(SHA3-512(x))`.

Against a quantum computer, Grover's algorithm speeds up the search for a preimage, but only quadratically. With 512 output bits, the remaining security is about 256 bits, far beyond anything reachable.

Each use of a hash is prefixed with its own label ("domain separation"), for example `NFR-SIGHASH`, `NFR-MULTISIG` or `NFR-wallet-key`. The same data therefore cannot be reinterpreted in another context.

3.2 Transaction signatures: ML-DSA-87

Every spend is authorised by an **ML-DSA-87** signature, the strongest parameter set of FIPS 204 (NIST category 5, comparable to AES-256). Signatures use the context `NFR-tx-v1`, which makes them unusable outside NumFranc.

	ML-DSA-87	ECDSA (Bitcoin)
Public key	2,592 bytes	33 bytes
Signature	4,627 bytes	71 to 73 bytes
Mathematical problem	lattices	discrete logarithm
Known quantum attack	none efficient	Shor's algorithm

This choice has a cost: a NumFranc transaction is roughly thirty times larger than a Bitcoin transaction. NumFranc accepts it and offsets it with larger blocks (section 5.4).

ML-DSA is a **stateless** scheme: signing several times with the same key does not weaken it. An address can therefore be reused without cryptographic risk.

3.3 Release signatures: SLH-DSA

Published releases of the software are signed with **SLH-DSA-SHAKE-128s** (FIPS 205). Its security rests only on hash functions, the most studied and most conservative assumption in cryptography. The signature is large (7,856 bytes) and slow to produce, but it is computed only once per release. Section 11 describes how it is used.

3.4 Proof of work: Argon2d

The proof of work uses **Argon2d** (RFC 9106), winner of the Password Hashing Competition, with 4 MiB of memory, one pass and a 64-byte output. Each attempt has to fill and then read back 4 MiB of memory in an unpredictable order. Specialised chips (ASICs) and graphics cards thus lose most of their edge over an ordinary processor, which keeps mining open to everyone.

3.5 Wallet protection

The wallet file is protected by an **Argon2id** key derivation, slow and memory-hard, then by **two successive authenticated encryptions**: AES-256-GCM, then XChaCha20-Poly1305 over the result, with two independent 256-bit keys. If one of the two algorithms were ever broken, the other would still protect the data. Details are in section 9.

3.6 Summary of quantum resistance

Component	Algorithm	Effect of a quantum computer
Transaction signatures	ML-DSA-87	no known efficient attack
Release signatures	SLH-DSA-SHAKE-128s	no known efficient attack
Hashes, Merkle, addresses	SHA3-512	Grover: security halved in bits, still ≥ 128 bits
Wallet encryption	AES-256 and XChaCha20	Grover: about 128 bits left

"No known attack" does not mean "proven unbreakable". That is why NumFranc uses the strongest parameter sets available, and a signature based on hashing alone for the most critical element: the authenticity of the software.

4. Monetary issuance

4.1 Total supply

There will never be more than **69,100,000 NFR**. This number is inspired by the population of France on 1 January 2026, estimated by INSEE at about 69.1 million inhabitants: symbolically, one NFR per inhabitant. The number 691 also appears in the time between blocks (691 seconds) and in the difficulty adjustment window (691 blocks).

All amounts are **integers** expressed in minimal units (1 NFR = 10^8 units). No consensus computation uses floating-point numbers: every node gets exactly the same result, on any computer.

4.2 Schedule in tiers

Issuance is divided into **tiers of 100,000 blocks**, i.e. 69.1 million seconds: about **2 years and 2 months** per tier.

- **Tier 0 (slow start)**: 6.91 NFR per block. This tier gives the network time to form before most coins are issued.
- **Tier 1 (peak)**: 69.10 NFR per block.
- **Following tiers**: the reward drops by **10%** at each tier, using integer division (`r ← r × 9 / 10`).

The last rewarded block is block **4,471,948**, which receives exactly the remainder (0.44971536 NFR). Issuance ends about **98 years** after the genesis block.

Tier	Blocks	Reward per block	Issued at end of tier	Years elapsed (approx.)
0	0 – 99,999	6.91 NFR	691,000	2.2
1	100,000 – 199,999	69.10 NFR	7,601,000	4.4
2	200,000 – 299,999	62.19 NFR	13,820,000	6.6
3	300,000 – 399,999	55.971 NFR	19,417,100	8.8
5	500,000 – 599,999	45.33651 NFR	28,988,141	13.1
10	1,000,000 – 1,099,999	26.77075578 NFR	45,697,319.79	24.1
44	4,400,000 – 4,471,948	0.74457068 NFR, then remainder	69,100,000	97.9

Appendix E gives the detailed schedule. Some milestones:

Share of total supply issued	Reached around block	Years after genesis
25%	361,728	7.9
50%	740,345	16.2
75%	1,379,416	30.2
90%	2,195,221	48.1
99%	3,813,590	83.5

4.3 Mining reward

The miner of a block is entitled to that block's reward **plus** the fees of the transactions it includes. The reward transaction ("coinbase") cannot exceed this total. A miner who claims less loses the difference: those units are never created. The reward can only be spent after **100 confirmations**.

The genesis block reward (6.91 NFR) is paid to a null address and cannot be spent, like that of Bitcoin's genesis block.

4.4 Distribution

There is **no pre-sale, no founders' reserve and no coins created outside mining**. Every NFR is issued by a publicly mined block, following the schedule above. Tier 0, with its low reward, limits the share of the very first miners, including the project author's.

5. Blocks and proof of work

5.1 Block header

Each block starts with a fixed-size header of **212 bytes**, written in big-endian order:

Field	Size	Role
Version	4 bytes	protocol version (currently 4)
Previous block	64 bytes	hash of the previous block's header
Merkle root	64 bytes	summary of all the block's transactions
Time	8 bytes	Unix time, in seconds
Target	64 bytes	difficulty threshold (512-bit integer)
Nonce	8 bytes	counter changed by the miner

The **block hash** is `H(header)`, the double SHA3-512. The **proof of work** is a separate computation: `Argon2d(header)`, with the fixed salt `NumFranc-PoW-v1!`. A block is valid if this value, read as a 512-bit integer, is **below the target**.

Transactions are summarised by a **Merkle tree** with domain separation: each leaf is `H(0x00 ‖ txid)`, each node `H(0x01 ‖ left ‖ right)`. This distinction prevents an internal node from passing as a transaction.

5.2 Genesis block

The first block is written into the software. It is dated **17 October 2026 at 00:00:00 UTC** and carries the message `NumFranc - 69,1 millions - 17/10/2026`. Its nonce is 452,924 and its hash starts with `7e0e17a3e283a029`. Its target, $2^{492} - 1$, requires on average about one million Argon2 attempts: a few minutes for a desktop computer.

5.3 Difficulty adjustment

Every **691 blocks**, the target is recalculated from the time that actually elapsed:

```
new target = old target × measured time / expected time
```

The expected time covers the window's 690 intervals: 690 × 691 = 476,790 seconds. The measured time is clamped between a quarter and four times this value, so that a single adjustment cannot change the difficulty by more than a factor of 4. The whole computation uses big integers, with no floating-point rounding.

Since version 1.0, a rule taken from Bitcoin (BIP94) prevents the so-called “time warp” attack: the first block of a window cannot be dated more than 2 hours before the previous block. Without it, a majority miner could date that block in the past to distort the measurement and lower the difficulty. Like the other rules added in 1.0, it applies to blocks whose median time past is after 19 October 2026: older blocks remain valid.

5.4 Block size and throughput

A block can hold up to **8 MiB**. A simple transaction, with one input and two outputs, takes about 7.4 kB, because of the size of ML-DSA keys and signatures. A block can therefore hold about 1,100 such transactions, for a throughput of about **1.6 transactions per second**. That is lower than Bitcoin: it is the price of post-quantum security. Improving throughput is one of the research directions (section 12).

5.5 Cumulative work

The work of a block is `2⁵¹² / (target + 1)`. The chain a node follows is the one with the greatest **cumulative work**, not the longest. An attacker therefore cannot impose a longer chain mined at an artificially low difficulty.

6. Transactions

6.1 The unspent output model

As in Bitcoin, coins are **unspent transaction outputs**. A transaction consumes existing outputs (its inputs) and creates new ones (its outputs). The sum of inputs must be strictly greater than the sum of outputs: the difference is the fee, paid to the miner.

NumFranc has **no scripting language**. All spending rules are described by fixed fields and checked by simple code. This greatly reduces the room for error, at the cost of less flexibility than Bitcoin.

6.2 Structure

Element	Content
Version, global lock	transaction version (4); the global lock is not used by the current rules
Inputs	spent output (txid, index), then either an ML-DSA public key and signature, or a multi-signature proof
Outputs	amount, address, unlock lock, optional cancel address
Attached message	UTF-8 text of at most 140 bytes
Nickname	optional registration or transfer of a nickname

A transaction's identifier is `H(full transaction)`. The signature covers a hash `H("NFR-SIGHASH" ‖ transaction)` in which keys and signatures are replaced by zeros. The attached message, the locks and the nickname are thus signed with the rest: nobody can change them in transit. The byte-level format is given in Appendix C.

6.3 Addresses

An address is the **first 256 bits of SHA3-512(public key)**. It is displayed in a readable form:

`NFR1-XXXX-XXXX-…`

The body encodes, in Crockford base 32, the address's 32 bytes followed by 4 checksum bytes (58 characters in groups of 4). This alphabet excludes I, L, O and U; when reading, I and L are understood as 1, and O as 0. A typo is detected by the checksum before anything is sent.

6.4 Fees

The minimum fee is **1,000 units per kilobyte started**. A simple 7.4 kB transaction therefore costs at least 8,000 units, i.e. 0.00008 NFR. Miners pick first the transactions that pay the most per byte.

6.5 Time-locked payments

Each output can carry an **unlock lock**. If it is 0, the output can be spent immediately. If it is below 500,000,000, it is a block height. Otherwise, it is a Unix date, compared with the **median time past** of the last 11 blocks, as in Bitcoin's BIP 113. A miner therefore cannot move the clock forward to unlock funds earlier.

Typical use: pocket money paid now but available only next month.

6.6 Retained (cancellable) payments

A locked output can also name a **cancel address**. Until the unlock date, only that address can take the funds back. After it, only the recipient can spend them. This is the basis of the wallet's **large-payment protection** (section 9.5): in case of a mistake or theft, the sender keeps a window to cancel.

6.7 Multi-signature

An address can belong to a **group of 2 to 15 members**, at least **M** of whom must sign to spend. The group's address is the hash of the threshold and of the sorted list of member addresses. The threshold and member list are covered by the signatures; the signatures themselves are not, which lets members sign one after another.

6.8 Nicknames

A transaction can **register a nickname**, for example `alice.nfr`, linked to an address. One can then pay that nickname instead of copying an address. A nickname has 3 to 32 characters from `a-z`, `0-9` and the hyphen (never at the start, at the end, or doubled). Only its current owner can transfer it.

To prevent hoarding of short names, registration costs an additional fee, paid to the miner:

Nickname length	Additional fee
3 characters	100 NFR
4 characters	25 NFR
5 to 7 characters	5 NFR
8 characters or more	1 NFR

7. Consensus and chain security

7.1 Validation rules

Every node checks every block it receives by itself. A block is rejected if any of the following rules is broken:

- correct protocol version, and consistency with the checkpoints written into the software;
- target equal to the one required by the difficulty adjustment;
- time strictly later than the median time of the previous 11 blocks, and at most **2 hours** in the future;
- Argon2d proof of work below the target;
- size of at most 8 MiB;
- first transaction, and only that one, is a coinbase containing the block height;
- exact Merkle root, and no duplicate transaction;
- every transaction valid: ML-DSA-87 signatures, existing unspent inputs, locks respected, sufficient fees, valid nicknames;
- coinbase not exceeding the reward plus fees.

Cheap checks come first; the proof of work (about 2 ms) and the signatures come next. A rejected block is never relayed to other nodes.

7.2 Protections against 51% attacks

An attacker controlling most of the mining power could try to rewrite recent history, for example to cancel a payment they made. NumFranc combines five protections:

1. **Cumulative work**: only the chain that required the most computation counts.
2. **Reorganisation limit**: a node refuses to replace more than **10 blocks** of its active chain (about 1 h 55 min). Beyond that, history is considered final.
3. **Checkpoints**: the hashes of known blocks can be written into each new software release. This list is empty in version 1.0.
4. **Reward maturity**: 100 confirmations before a mining reward can be spent.
5. **Timestamp rules**: the median time past, the 2-hour limit and the anti “time warp” rule (section 5.3) prevent manipulating the difficulty by cheating on dates.
6. **Dead branches rejected**: a block from a branch that starts more than 10 blocks back, and could therefore never be adopted, is not even stored.

The reorganisation limit has a trade-off worth knowing: if the network were split in two for more than 10 blocks, the two halves would not reconcile automatically. Manual intervention would then be needed. NumFranc considers this risk preferable to that of a deep rewrite of history.

For a large payment, it is advisable to wait for **6 confirmations**, about 70 minutes.

8. Network

8.1 Protocol

Nodes communicate over TCP on **port 6910**. Each message is framed as follows:

`identifier "NFR\x04" (4 bytes) | command (12) | length (4) | checksum (4) | payload`

The checksum is the first 4 bytes of `H(payload)`. A payload cannot exceed 8 MiB + 1 KiB. The commands are: `version`, `verack`, `getblocks`, `block`, `tx`, `tip`, `getaddr`, `addr`, `ping` and `pong` (Appendix F). A node opens at most 8 outgoing connections and accepts at most 32 incoming ones.

8.2 Finding other nodes without a central server

NumFranc depends on no server. A node finds its peers in four ways:

1. **On the local network**: a UDP announcement every 10 seconds on port 6910.
2. **Through BitTorrent's decentralised directory** (the "Mainline" DHT, used by millions of computers): each node registers under an identifier specific to NumFranc, derived from SHA3-512 and the genesis hash, then looks up the other registered nodes. The lookup runs every minute at startup, then every 10 minutes.
3. **By opening the port automatically** on the home router (UPnP, then NAT-PMP if UPnP does not answer), to be reachable from outside. The request is renewed every 10 minutes and repeated every 5 minutes while it fails; if the program stops, the router closes the port by itself within 20 minutes. The status (open, or why not and what to do) is shown on the home screen and in Settings > Connection. Without the open port, the node still works: it connects to others, but others cannot connect to it first.
4. **From memory** of peers already met, stored in the file `peers.txt`.

At startup, a node waits 20 seconds to synchronise before starting to mine, so as not to work on an outdated chain. It requests missing blocks in batches of at most 500.

8.3 Transaction pool

Valid transactions not yet included in a block wait in a pool of at most 64 MiB. Only one nickname registration at a time may target a given name there: this prevents a miner from producing a block with two conflicting registrations. When the pool is full, as in Bitcoin, a transaction that pays a higher fee per byte replaces the cheapest ones instead of being refused.

8.4 Protections against abuse

An open network inevitably receives messages from malicious peers. Since version 1.0, following a security audit, NumFranc adopts Bitcoin's protections:

- **Proof before trust**: a node pauses mining to synchronise only if a peer has recently **delivered** a valid block. A mere, unverifiable height announcement is not enough.
- **Blocks sent one by one**: in reply to a request, blocks are read and sent one at a time, up to 500 blocks or 32 MiB. Requests that come too close together are ignored, and an overly insistent peer is disconnected.

- **Cheapest checks first**: a relayed transaction must not exceed 2 MiB; the node then checks that the spent coins exist, and only at the end the signatures, outside the locks. A wrong signature gets the peer banned for 24 hours.
- **Two-table peer directory**, modelled on Bitcoin's *addrman*: addresses already tried and new ones. Addresses are grouped by Internet neighbourhood (the first two numbers of the IPv4 address): at most 8 new and 4 tried per neighbourhood, at most 64 brought by the same peer, and random replacement when a table is full. Private addresses (router, local network) coming from the Internet are rejected: computers at home find each other through local discovery. A node opens at most one outbound connection per neighbourhood and accepts at most 4 inbound connections from the same neighbourhood. A peer may send only one address every 10 seconds, on top of the reply to our own request.
- **Honest peers are not punished** (version 1.1): a block dated more than 2 hours in the future by our clock is ignored without banning its sender (either computer's clock may be off), and the program tells the user to check the time. A block already rejected, or built on a rejected block, only gets the connection closed, without a 24-hour ban. More than 20 orphan blocks (unknown parent) per minute from one peer also close the connection.
- **A slow peer delays nobody**: blocks and transactions are sent to each peer in parallel, so one overloaded connection no longer holds up the others.
- **No freezing while spending**: the heavy computation of the spending code (Argon2, several seconds) is done before taking any lock, so the chain, mining and the window keep running while a payment is prepared.

9. Wallet

9.1 A deterministic wallet

All keys of a wallet come from a **single 32-byte secret seed**. Key number i is obtained as follows:

```
key_i = ML-DSA-87.keygen(SHA3-512("NFR-wallet-key" ‖ seed ‖ i)[0..32])
```

A pool of 1,000 addresses is computed in advance. Receiving money and mining therefore never require the seed. The wallet keeps a permanent **main address** for receiving; each mining reward and each change output goes to a fresh address.

9.2 Two passwords, two levels

The wallet file contains two separately encrypted sections:

Section	Opened by	Allows	Cost of one guess (Argon2id)
The view (addresses, address book)	opening password, at least 12 characters	see the balance, receive, mine	256 MiB, 3 passes (about 1 s)
The vault (the seed)	spending code, at least 16 characters	send money	1 GiB, 4 passes (about 3 to 5 s)

Each section is encrypted twice (AES-256-GCM then XChaCha20-Poly1305) with a 512-bit key derived by Argon2id. The section's name is authenticated: the view and the vault cannot be swapped. A thief who copies the file must pay several seconds of computation and 1 GiB of memory **for every password tried**.

9.3 Recovery code in three pieces

At creation, the seed is split using **Shamir secret sharing** (2 of 3, over the finite field GF(256)) into three pieces written `NFRS-…`. **Any two** of them rebuild the seed. A single piece reveals nothing. One can thus keep one piece at home, a second with a relative and a third in another place: losing or having one piece stolen loses nothing.

9.4 Freezing

Freezing removes the seed from the computer; only a fingerprint of it remains. Even with both passwords, nothing can be sent from that computer any more, but receiving and mining continue. Unfreezing requires the opening password, the spending code and **two pieces** of the recovery code.

9.5 Large-payment protection

The user can set a threshold. Above it, each payment is sent as a **retained payment** (section 6.6) for a chosen delay, during which it can be cancelled. An optional **guardian**, i.e. another wallet or a multi-signature group, can also cancel it. A thief who obtained the passwords could therefore not empty the wallet instantly.

9.6 Everyday features

- **Visual fingerprint**: each address is associated with 4 simple words and a colour, to check at a glance that you are paying the right person.
- **Attached message**: up to 140 bytes of text, public and signed with the payment.
- **Address book** and payment to a **nickname**.
- **Payment links and QR codes**: `nfr:NFR1-…?amount=12.5&message=Thanks`. On Windows and Linux, clicking the link opens NumFranc with the send form already filled in; nothing is paid without checking and without the spending code.
- **Proof of reserve** (`NFRPROOF1-…`): prove you hold at least a given amount, revealing only the addresses needed and including a word chosen by the verifier, to prevent an old proof from being reused.
- **Multi-signature proposals** (`NFRMS1-…`): a transaction passed around the members of a group until the threshold is reached.

10. Application and distribution

NumFranc is **a single program**. A double-click opens a window that creates or opens the wallet, then starts the node and mining. The window shows the essentials: Home, Send, Receive, History, Address book and Settings, plus Security and Explain in a “More” menu. **Expert mode** (in Settings) adds Explorer, Issuance, Advanced and Network. The interface is available in **French and English**, with a light or dark theme. On first launch the program adds itself to the applications menu and

registers the `nfr:` payment links, by itself and only inside the user's home folder. Since version 1.1 it is distributed as **double-click installers**: a `.dmg` disk for the Mac (drag NumFranc onto Applications), an installer `.exe` for Windows (no administrator rights needed, uninstalling never touches the wallet) and `.deb` packages for Linux (amd64 and arm64). They are not signed by Apple or Microsoft, which requires paid accounts tied to a real identity: the first opening therefore shows one warning, explained on the download page. The hashes of the installers are listed in the same signed file `SHA3SUMS.txt`, so the release signature covers them too.

The software is written in **Rust**, a language that rules out whole families of memory errors by design. It is published under the **MIT licence** on GitHub. For each release, a continuous integration service runs the test suite, then builds the program for five platforms: Linux x86-64, Linux ARM64, macOS Apple Silicon, macOS Intel and Windows x86-64. It also produces the archive `NumFranc.zip` and the hash file `SHA3SUMS.txt`. The build is **reproducible**: the Rust version, the dependencies and the paths are fixed, so two builds of the same code give exactly the same program. The service checks this by building the Linux version twice; anyone can repeat the test with `scripts/build-reproductible.sh`.

11. Release signing

11.1 The problem

An attacker could publish a fake version of NumFranc, for example to steal wallets. The `SHA3SUMS.txt` hashes detect a damaged file, but not a complete forgery: the attacker could publish their own hashes.

11.2 The solution

Each published release comes with `SHA3SUMS.txt.sig`, an **SLH-DSA-SHAKE-128s** signature of the hash file. Only the holder of the secret signing key can produce it. Anyone can check it with the **public key** (32 bytes), published in the repository:

```
5e1d1c80fdd07d3bc4095f45a05649fc006bce347609b1566b3c25e940881736
```

Verification uses the provided tool:

```
signer_version verifier SHA3SUMS.txt SHA3SUMS.txt.sig cle-publique.txt
```

If the signature is wrong, the tool prints "signature FAUSSE : ne pas utiliser ce fichier" (signature WRONG: do not use this file). This whitepaper itself is included in the signed files.

The **“Check for updates”** button (Settings) performs the same check automatically: it downloads `SHA3SUMS.txt` and its signature, verifies the signature with the public key built into the program, and only offers the new version if it is valid. The version number is read from the first line of the hash file, which is also covered by the signature.

11.3 Protecting the signing key

The secret key was created on a computer disconnected from the network. It no longer exists in full anywhere: it was split by Shamir sharing (two pieces out of three are enough), each piece was encrypted, then stored offline in a separate place. It is rebuilt only for the time needed to sign a

release, offline.

11.4 In case of compromise

If the signing key were ever compromised, a new key pair would be created, the new public key would be published in the repository with an explicit notice, and releases signed after the date of compromise would be declared untrustworthy. Releases signed before would remain valid.

12. Roadmap

The following items are **not yet implemented**. They are listed without dates, in approximate order of priority:

- **Security audit by human experts** of the consensus code and the wallet. A first audit, carried out by an independent artificial intelligence, led to the protections of section 8.4.
- **Public bootstrap nodes**: always-on nodes to ease the first connection. The current discovery (DHT, local network, UPnP) works without them.
- **Dedicated signing computer**, never connected, for releases.
- **Local RPC interface** (`127.0.0.1:6911`), reachable only from the computer itself, for future exchange gateways. The address is reserved in the code but the service does not exist yet.
- **Headers-first synchronisation** (like Bitcoin): download the light headers first, then the blocks.
- **Checkpoints** added with each new release.
- **Throughput**: study of techniques to reduce the space taken by post-quantum signatures in blocks.

13. Conclusion

NumFranc applies Bitcoin's proven mechanisms with cryptography chosen to last: ML-DSA-87 signatures, SHA3-512 hashes, SLH-DSA-signed releases, and Argon2d mining open to everyone. Its issuance is fixed, entirely mined and public. Its wallet and transactions include concrete protections against mistakes and theft. The project is open: anyone can read the code, check it, run a node and propose improvements.

References

1. S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.
2. NIST, FIPS 204 — *Module-Lattice-Based Digital Signature Standard* (ML-DSA), August 2024.
3. NIST, FIPS 205 — *Stateless Hash-Based Digital Signature Standard* (SLH-DSA), August 2024.
4. NIST, FIPS 202 — *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, 2015.
5. A. Biryukov, D. Dinu, D. Khovratovich, S. Josefsson, RFC 9106 — *Argon2 Memory-Hard Function*, IETF, 2021.
6. P. Shor, *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*, 1994.

7. L. Grover, *A Fast Quantum Mechanical Algorithm for Database Search*, 1996.
8. A. Shamir, *How to Share a Secret*, Communications of the ACM, 1979.
9. Y. Nir, A. Langley, *RFC 8439 — ChaCha20 and Poly1305 for IETF Protocols*, IETF, 2018.
10. NIST, *SP 800-38D — Galois/Counter Mode (GCM)*, 2007.
11. Bitcoin, *BIP 113 — Median time-past as endpoint for lock-time calculations*, 2015.
12. BitTorrent, *BEP 5 — DHT Protocol*.
13. D. Crockford, *Base 32 Encoding*.
14. INSEE, *Population au 1er janvier — Données annuelles de 1990 à 2026*.

Technical appendices

These appendices describe NumFranc precisely enough for another programmer to check the software or write a compatible implementation. In case of any difference, the source code of the corresponding release prevails.

Appendix A — Consensus and network parameters

Constant	Value	Meaning
`MIN_UNIT`	100,000,000	units per NFR
`HARD_CAP`	6,910,000,000,000,000 units	69,100,000 NFR
`PALIER_BLOCKS`	100,000	blocks per issuance tier
`SLOW_START_REWARD`	691,000,000	6.91 NFR, tier 0
`PEAK_REWARD`	6,910,000,000	69.10 NFR, tier 1
`TARGET_BLOCK_TIME`	691 s	target time between blocks
`DIFFICULTY_WINDOW`	691 blocks	adjustment window
`EXPECTED_TIMESPAN`	476,790 s	690 × 691
`MTP_BLOCKS`	11	blocks in the median time past
`MAX_FUTURE_DRIFT`	7,200 s	maximum time ahead
`HEADER_SIZE`	212 bytes	block header
`MAX_BLOCK_SIZE`	8,388,608 bytes	8 MiB
`MAX_TX_SIZE`	8,384,512 bytes	8 MiB − 4,096
`PROTOCOL_VERSION`	4	block version
`TX_VERSION`	4	transaction version
`MAX_MEMO`	140 bytes	attached message
`MIN_FEE_PER_KB`	1,000 units	per kilobyte (1,000 bytes) started
`COINBASE_MATURITY`	100 blocks	reward maturity
`MAX_REORG_DEPTH`	10 blocks	maximum reorganisation
`SAFE_CONFIRMATIONS`	6	recommended confirmations
`MAX_TARGET`	$2^{512} - 1$	easiest target
`GENESIS_TARGET`	$2^{492} - 1$	initial target
`GENESIS_TIMESTAMP`	1,792,195,200	17/10/2026 00:00:00 UTC
`GENESIS_NONCE`	452,924	genesis nonce

Constant	Value	Meaning
`POW_MEMORY_KIB`	4,096	Argon2d memory (KiB)
`POW_SALT`	`NumFranc-PoW-v1!`	proof-of-work salt
`LOCK_TIME_THRESHOLD`	500,000,000	height or date
`MAX_MULTISIG_MEMBERS`	15	group members
`NAME_MIN_LEN` / `NAME_MAX_LEN`	3 / 32	nickname length
`SIG_CONTEXT`	`NFR-tx-v1`	ML-DSA context
`P2P_PORT`	6910	TCP and UDP port
`MAGIC`	`4E 46 52 04`	"NFR" and version 4
`MAX_OUTBOUND` / `MAX_INBOUND`	8 / 32	connections
`MAX_BLOCKS_PER_REPLY`	500	blocks per reply
`STARTUP_SYNC_SECS`	20 s	wait before mining

Full genesis block hash:

`7e0e17a3e283a0291917d8f8c04605bbd5ab00b8e0459baafc0dd0d698238c74`
`fb5fbc288403453f5f3ad2aef5aed636b367f8b654e6bd5e7327360257cbf79e`

Appendix B — Proof-of-work and difficulty computations

- Block hash: `H(header) = SHA3-512(SHA3-512(212-byte header))`.
- Proof of work: `Argon2d(password = header, salt = "NumFranc-PoW-v1!", m = 4,096 KiB, t = 1, p = 1, output = 64 bytes)`, version 0x13.
- Validity: the Argon2d output, read as a 512-bit big-endian integer, is strictly below the target.
- Block work: $\lfloor 2^{512} / (\text{target} + 1) \rfloor$.
- Adjustment, at every height h that is a multiple of 691: $T = \text{time}(h-1) - \text{time}(h-691)$, clamped to $[119{,}197,\ 1{,}907{,}160]$; $\text{new target} = \lfloor \text{old target} \times T / 476{,}790 \rfloor$, clamped to $[1,\ 2^{512} - 1]$.
- At other heights, the target is the parent block's target.

Appendix C — Binary transaction format

All integers are big-endian. A "varint" is a variable-size integer, in mandatory canonical form.

Field	Size
version	4
global lock (not used by the current rules)	4
number of inputs	varint

Field	Size
for each input: previous txid	64
previous output index	4
— coinbase (null txid, index 0xFFFFFFFF): data length	1
— coinbase: data (8 bytes of height, then 0 to 92 free bytes)	8 to 100
— simple input: type = 0, ML-DSA public key, signature	1 + 2,592 + 4,627
— multi-signature input: type = 1, M, N, N addresses	1 + 1 + 1 + 32 × N
— then number of signatures K, and K times (index, public key, signature)	1 + K × (1 + 2,592 + 4,627)
number of outputs	varint
for each output: amount	8
address	32
unlock lock	4
cancel flag (0 or 1), then cancel address if 1	1 (+ 32)
attached message length	varint
attached message (UTF-8)	0 to 140
nickname flag (0 or 1)	1
if 1: type (0 = register, 1 = transfer), length, name, owner address	1 + 1 + n + 32

Identifiers and signatures:

- `txid = H(serialised transaction)`;
- `sighash = H("NFR-SIGHASH" ‖ transaction)`, where keys and signatures of simple inputs are replaced by zeros of the same size, and where, for multi-signature inputs, the threshold and member list are kept but the signatures removed;
- simple address: `SHA3-512(public key)[0..32]`;
- group address: `SHA3-512("NFR-MULTISIG" ‖ M ‖ N ‖ sorted addresses)[0..32]`;
- minimum fee: `⌈size / 1,000⌉ × 1,000` units, plus the nickname fee where applicable.

Spending rule for an output: if its lock is released, only the recipient can spend it; otherwise only the cancel address can, if there is one. A zero lock is always released; a lock below 500,000,000 is released from that height; otherwise it is released when the median time past reaches that date.

Appendix D — Text formats

Object	Form	Detail
Address	`NFR1-XXXX-…`	Crockford base 32 of (address ‖ `H(address)[0..4]`), 58 characters in groups of 4

Object	Form	Detail
Recovery piece	`NFRS-…`	Shamir share of the seed, with checksum
Payment link	`nfr:<address or nickname>?amount=…&message=…`	readable as a QR code
Proof of reserve	`NFRPROOF1-<hex>`	revealed addresses, signatures and challenge word
Multi-signature proposal	`NFRMS1-<hex>`	serialised transaction to complete
Nickname	`name.nfr`	3 to 32 characters `a-z 0-9 -`

Appendix E — Issuance schedule

Reward per block at the start of each tier, in NFR. Each tier has 100,000 blocks, about 2.19 years.

Tier	Reward	Issued before the tier	Start (years)
0	6.91	0	0.0
1	69.10	691,000	2.2
2	62.19	7,601,000	4.4
3	55.971	13,820,000	6.6
4	50.3739	19,417,100	8.8
5	45.33651	24,454,490	11.0
6	40.802859	28,988,141	13.1
7	36.7225731	33,068,426.9	15.3
8	33.05031579	36,740,684.21	17.5
9	29.74528421	40,045,715.79	19.7
10	26.77075578	43,020,244.21	21.9
11	24.0936802	45,697,319.79	24.1

Later tiers continue the 10% drop per tier, using integer division on units. The total reaches exactly 6,910,000,000,000,000 units at block 4,471,948, in tier 44. From block 4,471,949 on, the reward is zero and miners are paid only by fees.

Appendix F — Network messages

Command	Payload	Role
`version`	version, height, listening port, nonce	introduction when a connection opens
`verack`	empty	acknowledges `version`

Command	Payload	Role
`getblocks`	list of known hashes	request for missing blocks
`block`	a full block	sends a block
`tx`	a transaction	relays a transaction
`tip`	a height	announces the height of one's chain
`getaddr`	empty	request for peer addresses
`addr`	list of addresses	reply to `getaddr`
`ping` / `pong`	nonce	checks the connection is alive

The `version` nonce lets a node detect that it has connected to itself.

Appendix G — Glossary

- **Block**: a bundle of transactions added to the chain by a miner.
- **Coinbase**: the first transaction of a block, which creates the miner's reward.
- **Confirmation**: number of blocks added after the one containing a transaction.
- **Hash**: fixed-size summary of data; the slightest change alters the whole summary.
- **Genesis**: the very first block, written into the software.
- **Seed**: 32-byte secret from which all of a wallet's keys come.
- **Mempool**: pool of transactions waiting for a block.
- **Nonce**: number the miner varies to find a valid proof of work.
- **Node**: a computer running NumFranc and checking the chain.
- **Post-quantum**: describes an algorithm designed to withstand quantum computers.
- **Proof of work**: a computation that is expensive to produce but quick to check, which protects the chain.
- **Reorganisation**: replacing the last blocks of the chain with a competing chain that has more work.
- **Median time past (MTP)**: median of the times of the last 11 blocks.
- **UTXO**: unspent transaction output; these are NumFranc's "coins".

NumFranc's code has not yet undergone an independent security audit. No sale of NFR is offered and no value is promised. This document describes free software; it is neither an offer nor financial advice.