NodeFlow

DOCUMENT:	TECHNICAL DISCLOSURE
AUTHOR:	JARET WALKER
ORIGINAL PUBLICATION:	APRIL 4, 2026
UPDATED:	JULY 21, 2026

NODEFLOW

Negentron - deterministic node identity generator

Bondatron - system to node bonding mechanism

Veritron - deterministic verification and continuity advancement

Movitron - nested encryption and transport

Membrane - public verification boundary

Lumen - internal payload processing

NEGENTRON

```
private_key = derive_key(seed_material)

public_key = compress(
    public(private_key)
)

node_id = HASH(public_key)
```

BONDATRON

```
bond = sign(
    system_private_key,
    node_id || public_key
)

verify(
    system_public_key,
    node_id || public_key,
    bond
)
```

VERITRON

BUILD

```
payload_hash = HASH(
    prev_hash ||
    payload ||
    timestamp
)

payload_signature = sign(
    private_key,
    payload_hash
)
```

VERIFY

```
HASH(public_key) == node_id

verify(
    system_public_key,
    node_id || public_key,
    bond
)

verify(
    public_key,
    payload_hash,
    payload_signature
)

payload_hash == HASH(
    prev_hash ||
    payload ||
    timestamp
)

prev_hash ==
current last_hash
```

CONTINUITY

```
last_hash = payload_hash
```

RESYNC

```
challenge = HASH(
    node_id ||
    context
)

challenge_signature = sign(
    private_key,
    node_id || challenge
)

verify(
    public_key,
    node_id || challenge,
    challenge_signature
)

prev_hash =
current last_hash
```

MOVITRON

NODE TO MEMBRANE

```
payload.body =
encrypt_for_lumen(
    payload.body
)

inner_object = {
    node_id,
    public_key,
    bond,
    prev_hash,
    payload_hash,
    payload_signature,
    timestamp,
    payload,
    metadata
}

outer_request = {
    sender_public_key,
    nonce,
    ciphertext
}

ciphertext =
encrypt_for_membrane(
    inner_object
)
```

MEMBRANE

```
inner_object =
decrypt_for_membrane(
    outer_request.ciphertext
)

sender_public_key ==
inner_object.public_key

verify identity
verify bond
require bonded == true
verify signature
verify payload hash
verify continuity

payload.body remains encrypted
```

MEMBRANE TO LUMEN

```
validated payload
    ->
internal Lumen processing
```

LUMEN

```
payload.body =
decrypt_for_lumen(
    payload.body
)

response_plaintext =
process(
    payload.body
)

response.body =
encrypt_body_for_node(
    response_plaintext
)
```

MEMBRANE TO NODE

```
response_object = {
    body,
    metadata
}

outer_response = {
    sender_public_key,
    nonce,
    ciphertext
}

ciphertext =
encrypt_transport_for_node(
    response_object
)

response returned over
the existing TCP connection
```

NODE

```
response_object =
decrypt_transport_response(
    outer_response.ciphertext
)

response.body =
decrypt_response_body(
    response.body
)
```

INNER OBJECT

node_id

public_key

bond

prev_hash

payload_hash

payload_signature

timestamp

metadata

payload

METADATA

host

port

RETAINED STATE

node_id
identifies the Node

last_hash
enforces continuity

bonded
permits individual Node disablement

TEMPORARY CHALLENGE STATE

node_id

challenge

issued_at

expires_at

STATEMENT OF DISCLOSURE

This document establishes a public technical description of the NodeFlow system as originally disclosed on April 4, 2026, and updated on July 21, 2026.

Equivalent cryptographic algorithms, key formats, encodings, or transport methods that preserve the same identity, bonding, verification, continuity, encryption, and trust boundary model are considered implementations of this system.

RECORDS

https://web.archive.org/web/20260309024414/https://silogic.foundation/

https://web.archive.org/web/20260322203038/https://silogic.foundation/

https://web.archive.org/web/20260405003918/https://silogic.foundation/