

TorUsNet 9.0 ENG

L'Architech (Gaberiel Breton-Breault)

1 General Introduction

1.1 — Foreword: Natural Evolution Toward TorUsNet 9.0

The protocol initially named *TorusNet* evolves in version 9.0 into a fully assumed identity: **TorUsNet**. This change reflects a major transition in the philosophy of the project.

In its early versions, TorusNet was designed as an independent directional protocol, demonstrating the feasibility of a communication architecture based on an **inviolable private torus** and an **opaque public torus**. With TorUsNet 9.0, the protocol reaches a decisive milestone: it now relies on a **volumetric directional architecture 100×100×100**, capable of internal mutation, dynamic rewriting, and three-dimensional evolution.

The name *TorUsNet* expresses this triple nature:

- **Tor**: continuity with the Tor browser ecosystem and its privacy objectives.
- **Us**: direct reference to the *torus*, the core of directional geometry.
- **Net**: a non-linear, volumetric, structurally inviolable network layer.

TorUsNet is **not** a fork of Tor. It is a **living directional architecture**, designed to replace the transport layer while preserving the spirit of the Tor browser: anonymity, confidentiality, resilience, and freedom of access.

Version 9.0 marks the beginning of a convergence: TorUsNet becomes a protocol designed for Tor, but also for photon-compatible architectures, the **VSMD project**, and the **SMD vessel**.

1.2 — Introduction of Directional Keys 9.0 (transmitted over the network)

TorUsNet 9.0 introduces a new type of directional key, optimized to operate with a **100×100×100 volumetric torus**. These keys are transmitted before the directional path and serve to properly initialize the private volumetric torus.

Structure of Directional Keys 9.0

Each key has the following structure:

- **20 bits — Starting point** Indicates the initial cell in the volumetric private torus where the interpretation of the path begins.

- **16 bits — Cell ID** Identifies the target cell in the private torus for this key.

Summary

Directional key = 20 bits (start) + 16 bits (ID) = 36 bits

Compact and fast to transmit, they allow the private torus to be initialized without exposing its three-dimensional internal geometry.

Role in the protocol

These keys:

- prepare the private torus to receive the full path,
- define the initial directional zone within the volume,
- ensure coherence between the private torus and the transmitted path,
- allow more precise reconstruction,
- guarantee correct interpretation of volumetric movements.

They are transmitted **before** the path because they serve as **directional pre-configuration**.

1.3 — Direct Transmission of Directional Paths (Model 9.0)

In previous versions, the protocol used a seed to generate the directional path. TorUsNet 9.0 introduces an extended alternative: **direct transmission of the complete volumetric directional path**.

From now on:

- the directional path can be sent directly over the network,
- the volumetric private torus uses it as-is to reconstruct the file,
- no internal information of the torus is exposed,
- security relies entirely on **3D directional geometry**.

The transmitted path contains:

- no internal key,
- no private torus structure,
- no directional geometry,
- no exploitable information.

It is only a **volumetric movement vector**, with no cryptographic value.

1.4 — Transmission of the Reconstructed File Hash

To ensure that the file reconstructed by the volumetric private torus is:

- complete,
- intact,
- uncorrupted,
- coherent with the transmitted path,

TorUsNet 9.0 introduces the transmission of the **hash of the reconstructed file**.

This hash:

- is sent over the network,
- reveals no directional information,
- allows no derivation of the torus,
- compromises no key,
- serves only to validate the integrity of the final file.

It becomes external proof that the directional reconstruction was performed correctly, even in cases of:

- lost keys,
- partially received paths,
- multiple reconstructions,
- network interruptions.

1.5 — Coexistence of Two Models: Seed or Transmitted Path

TorUsNet 9.0 still allows the seed to be completely replaced by a full path transmitted directly over the network. However, this direct transmission can increase network traffic.

For this reason, TorUsNet 9.0 maintains both models:

Model A — Directional Seed (compact, local)

- The 70-bit seed is transmitted.
- The volumetric private torus generates the path locally.

- Network traffic is minimal.
- Ideal for low-bandwidth environments.

Model B — Transmitted Directional Path (complete, external)

- The full path is sent over the network.
- The seed is no longer required.
- The protocol becomes simpler to implement.
- Ideal for integration into a Tor browser or photon-based architecture.

Model choice

The choice between seed or transmitted path must be made at programming time, depending on:

- system constraints,
- available bandwidth,
- desired simplicity,
- type of application,
- developer preferences.

Both models are officially supported and documented in **TorUsNet 9.0**.

Chapter 1 — Introduction to the Directional Architecture (TorUsNet 9.0)

1.1. General Vision

The directional architecture is an entirely new communication technology. It does not rely on mathematics, cryptographic derivations, or classical structures, but on a **living internal mechanism**, impossible to derive, impossible to explore, impossible to falsify.

TorUsNet 9.0 now relies on four fundamental elements, revised for the new volumetric architecture:

1. **The Volumetric Private Torus** — an internal structure of $100 \times 100 \times 100 = 1,000,000$ cells, each containing:
 - 1 normal bit,

- 1 implicit ghost bit (NOT of the normal bit),
- 4 possible directions. The private torus is unique to each session, never transmitted, never exported, never derivable.

2. **The Volumetric Directional Mechanics** — a set of rules defining:

- how a key moves inside the 3D torus,
- how IDs mutate along the three axes,
- how internal paths activate,
- how normal/ghost pairs invert,
- how internal coherence is maintained inside a volume.

3. **The Volumetric Glass Fortress** — an internal module that:

- validates 3D directional coherence,
- monitors extended directional ports,
- monitors internal IDs,
- destroys the session in case of anomaly,
- blocks hostile IPs. It is a structural directional firewall.

4. **Directional Keys (TorUsNet 9.0)** — a type of key sent before the path:

- 20 bits: starting point,
- 16 bits: cell ID, for a total of 36 bits per key. These keys prepare the volumetric private torus to receive the transmitted path.

The directional architecture does **not** use:

- encryption,
- mathematical derivation,
- signatures,
- certificates,
- private keys,
- cryptographic primitives.

It relies exclusively on:

- directional windows,
- directional constants,
- volumetric internal paths,
- ID mutations,
- internal paths,
- normal/ghost pairs,
- non-derivable internal structures.

1.2. Why a Directional Architecture?

Classical protocols (TLS, SSH, IPSec, WireGuard, etc.) rely on:

- mathematics,
- derivations,
- private keys,
- certificates,
- signatures.

These systems are vulnerable to:

- mathematical analysis,
- quantum computers,
- photonic computers,
- derivation attacks,
- collisions,
- replay attacks,
- injections.

The directional architecture eliminates these attack vectors entirely. It does not rely on any exploitable mathematical structure. It relies on an internal mechanism impossible to derive, impossible to explore, impossible to reverse — even with infinite computational power.

An attacker cannot:

- derive a volumetric private torus,
- explore a private torus,
- reconstruct a private torus,
- predict a directional path,
- falsify a directional key,
- repeat a directional key,
- reconstruct an internal ID.

1.3. The Three Fundamental Elements (Version 9.0)

1.3.1. The Volumetric Private Torus

The Private Torus is an internal structure of 1,000,000 cells, each containing:

- 1 normal bit,
- 1 implicit ghost bit,
- 4 possible directions.

It is:

- unique to each session,
- fractally reconstructed,
- never transmitted,
- never exported,
- never derivable.

It contains:

- internal IDs,
- the global ID,
- the unique torus ID,
- internal paths,
- normal/ghost pairs.

Inversion of normal/ghost bits during the path

In TorUsNet 9.0, each time a cell is traversed by the volumetric directional path, the normal and ghost bits automatically invert:

- normal = 0 $\rightarrow$ 1
- ghost = 1 $\rightarrow$ 0 or vice versa.

This inversion:

- creates directional oscillation,
- modifies the internal geometry of the 3D torus,
- prevents any prediction,
- reinforces non-derivability,
- makes each path unique and non-reproducible.

This rule remains fundamental in TorUsNet 9.0.

Volumetric directional mutation

New rule 9.0:

If the intensity crosses N layers, then the N layers mutate.

In a 100×100×100 torus:

- an intensity crosses 100 layers,
- triggers 100 mutations,
- loops back,
- crosses again,
- mutates again.

Mutation propagates along the three axes: X, Y, Z.

1.3.2. The Volumetric Directional Mechanics

It defines:

- how a key moves inside the 3D torus,
- how IDs mutate,
- how internal paths activate,

- how normal/ghost pairs invert,
- how internal coherence is maintained.

It is:

- non-linear,
- non-derivable,
- non-reversible,
- living,
- oscillating,
- mutating.

1.3.3. The Volumetric Glass Fortress

The Glass Fortress:

- validates 3D directional coherence,
- monitors directional ports,
- monitors internal IDs,
- destroys the session in case of anomaly,
- blocks hostile IPs.

It is the directional equivalent of a volumetric structural firewall.

1.4. Directional Ports: Extended Digital Ports

Directional ports are:

- classical digital ports,
- but with an extended range in the billions,
- generated by directional keys,
- validated by the Glass Fortress.

They are used to:

- transport directional transmissions,
- avoid collisions,

- prevent scans,
- prevent injections,
- prevent flow hijacking.

Examples:

- 1,482,993,112
- 7,000,221,991
- 3,991,000,000

Impossible to scan. Impossible to predict. Impossible to falsify.

1.5. A Post-Internet Architecture

The directional architecture is designed for the era:

- post-quantum,
- post-cryptographic,
- photonic.

It does not depend on:

- mathematics,
- certificates,
- private keys,
- cryptographic primitives.

It still uses:

- DNS,
- HTTPS,

but only as a **temporary entry gate**, before the session becomes **100% directional**.

1.6. A Directional Session: Living and Autonomous

A directional session is:

- living,
- autonomous,

- self-coherent,
- self-validating,
- self-mutating.

It evolves with each key:

- ID mutation,
- normal/ghost inversion,
- internal progression,
- update of the volumetric private torus,
- validation by the Glass Fortress.

It cannot be:

- copied,
- cloned,
- derived,
- predicted,
- falsified.

1.7. Objectives of the Directional Protocol

TorUsNet aims to:

1. Create sessions inviolable by design.
2. Replace classical cryptographic protocols.
3. Create private directional networks (multi-volumetric torus).
4. Enable directional file reconstruction.
5. Create an internal communication layer (directional ports).
6. Create a unique directional identity.
7. Ensure post-quantum and photonic security.
8. Extend the architecture to photonic systems, VSMD, and the SMD vessel.

1.8. Summary of Chapter 1

The directional architecture:

- is not cryptographic,
- is not mathematical,
- is not derivable,
- is not falsifiable.

It relies on:

- a 100×100×100 volumetric private torus,
- a living internal mechanics,
- 3D directional coherence,
- extended digital ports,
- a volumetric Glass Fortress,
- transmitted directional keys,
- a transmitted or seed-generated path,
- a final hash validating reconstruction,
- automatic inversion of normal/ghost bits,
- volumetric directional mutation through traversal.

TorUsNet 9.0 represents a new generation of protocol, designed for the post-quantum, photonic, and directional era.

Chapter 2 — Redesign of the Directional Seed and Introduction of the Transmitted Path Model (TorUsNet 9.0)

TorUsNet 9.0 introduces a major evolution in the way directional trajectories are generated, transmitted, and interpreted. Two models now coexist:

- The classical model based on the **directional seed (70 bits)**
- The modern model based on the **directional path transmitted directly over the network (9.0 model)**

Both models are officially supported and documented.

2.1 — Structure of the Directional Key (seed model, 128 bits)

The full directional key retains its original 128-bit structure, composed of:

- **2 bits — Key type**
- **20 bits — Directional starting point**
- **20 bits — Directional ending point**
- **16 bits — Target cell ID**
- **70 bits — Directional seed**

Total: 128 bits

This key remains compatible with the 100×100×100 volumetric torus because it defines only directional points, not the internal geometry.

2.2 — Structure of the Directional Seed (70 bits)

The directional seed is a compact 70-bit block containing five components:

- **Exponential base (4 bits)**
- **Fixed tower selection (4 bits)**
- **Fixed tower height (2 bits)**
- **Dynamic levels (48 bits)**
- **Final modulation (12 bits)**
 - Addition: 4 bits
 - Multiplication: 8 bits

This exponential engine remains unchanged in TorUsNet 9.0 because it is independent of volumetric geometry.

2.2.1 — Exponential Base (4 bits)

The 4 bits define a base among 16 hierarchical values:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 1000, 10,000, 100,000, 1,000,000, 10,000,000

2.2.2 — Fixed Tower Selection (4 bits)

Complete list of the 15 available fixed towers, each following the rule:

$$n^{\left((n+1)^{\left((n+2)^{\left((n+3)^{\dots}\right)}\right)}\right)}$$

where $n \in [1, 15]$.

The 4 bits allow:

- **0000** → no fixed tower
- **0001 to 1111** → selection of one fixed tower among the 15

2.2.3 — Fixed Tower Height (2 bits)

Determines the number of levels used:

- **00** → 1 level
- **01** → 2 levels
- **10** → 3 levels
- **11** → 4 levels

2.2.4 — Dynamic Levels (48 bits)

The 48 bits are divided into 6 dynamic levels, each represented by an 8-bit raw value:

$$D_1, D_2, D_3, D_4, D_5, D_6 \in [0, 255]$$

Directional rule:

- if a level = 0 → ignored
- if > 0 → active in the exponential tower

2.2.5 — Final Modulation (12 bits)

Final modulation consists of:

- **Addition (A)** — 4 bits → 0 to 15
- **Multiplication (M)** — 8 bits → 0 to 255

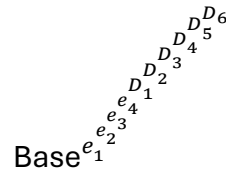
The final exponential value becomes:

$$E_{\text{final}} = (X + A) \times M$$

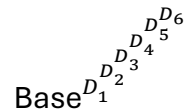
where **X** represents the complete tower.

2.3 — Final Exponential Structure

With fixed tower:



Without fixed tower:



Then modulation:

$$E_{\text{final}} = (X + A) \times M$$

2.4 — Representation Capacity

- **Raw space:**

$$2^{48}$$

- **Exponential space:** Far exceeds cryptographic, quantum, and photonic capabilities.

2.5 — Impact on the Volumetric Private Torus

The 100×100×100 volumetric private torus uses:

- the exponential base,
- the fixed tower (if activated),
- the 6 dynamic levels,
- the final modulation (A and M),
- the rule ignoring zero levels,

to determine the internal directional trajectory across the three axes X/Y/Z.

Normal/Ghost inversion

During the path:

- each traversed cell inverts its normal and ghost bits,
- modifying the internal geometry of the volumetric torus,

- making each path unique and non-reproducible.

Volumetric directional mutation (new in 9.0)

If intensity crosses **N layers**, the **N layers mutate**.

In a 100×100×100 torus:

- intensity crosses 100 layers,
- triggers 100 mutations,
- loops,
- crosses again,
- mutates again.

Mutation propagates through X, Y, Z.

2.6 — Introduction of the Transmitted Path Model (TorUsNet 9.0)

TorUsNet 9.0 introduces a second model:

The complete volumetric directional path is transmitted directly over the network.

This model:

- removes the need for the seed,
- simplifies directional generation,
- enables natural integration into a Tor browser,
- slightly increases network traffic,
- requires the new directional keys (36 bits).

Transmission order:

1. Directional keys (36 bits)
2. Complete volumetric directional path
3. Hash of the reconstructed file

2.7 — Conclusion of Chapter 2

TorUsNet 9.0 introduces:

- a 70-bit directional seed,

- a multi-level exponential engine,
- 15 fixed towers,
- configurable height,
- 6 dynamic levels (48 bits),
- final modulation (12 bits),
- a model based on transmitted paths,
- new directional keys (36 bits),
- transmission of the final hash,
- automatic inversion of normal/ghost bits,
- volumetric directional mutation through traversal,
- 3D directional coherence,
- a living and self-mutating architecture.

This structure reinforces the inviolable, non-derivable, and geometrically unpredictable nature of **TorUsNet 9.0**.

Chapter 3 — Glass Fortress (Structural Directional Firewall, TorUsNet 9.0)

The Glass Fortress is the **structural directional firewall** of the volumetric private torus. It validates, monitors, and protects all directional operations inside a 100×100×100 torus, according to the model chosen by the programmers:

- **Model A:** Transmitted directional key (36 bits)
- **Model B:** Full directional key (128 bits)

Both models are supported, but must **never** be used simultaneously. The Glass Fortress automatically adapts to the active model.

3.1. Overview

The Glass Fortress validates:

- the directional key (36 bits or 128 bits),
- the transmitted ID (16 bits),

- the reconstructed ID (16 bits),
- the fractal coherence of the volumetric torus,
- the directional reconstruction of the file,
- the normal/ghost inversion during traversal,
- volumetric directional mutation (new in 9.0).

It protects:

- the 100×100×100 private torus,
- the directional key of the active model,
- 3D directional coherence,
- file reconstruction,
- internal fractal stability.

It destroys the session in case of anomaly.

3.2. Fundamental Role

✓ Validation ✓ Monitoring ✓ Protection ✓ Destruction

The Glass Fortress is a **structural directional firewall**, designed for the post-quantum and photonic era.

3.3. Joint Validation: Glass Fortress + Host

Regardless of the chosen model (36 bits or 128 bits), the Glass Fortress **always** validates the directional key together with the Host.

Operation

1. The Host receives the directional key (36 or 128 bits).
2. The Host performs a preliminary directional validation.
3. The Glass Fortress performs the final structural validation.

If both validate the key:

The Glass Fortress authorizes:

- mutation of the internal 16 bits,

- internal reconstruction,
- execution of the volumetric path,
- normal/ghost inversion,
- directional mutation by traversal (100 mutated layers if 100 traversals occur).

If either rejects the key:

The Glass Fortress:

- blocks all mutation,
- blocks all reconstruction,
- destroys the session,
- erases the private torus,
- blocks the IP.

Absolute Directional Rule

The directional key must be validated by **both** the Host **and** the Glass Fortress.

3.4. Model A — Validation of the Transmitted Directional Key (36 bits)

This model is used when programmers choose to transmit the directional key over the network.

The Glass Fortress verifies:

1. The 20-bit starting point

- existing cell in the volume,
- 3D directional coherence,
- fractal coherence.

2. The 16-bit transmitted ID

- existing ID,
- non-corrupted ID,
- non-falsified ID.

3. The 16-bit reconstructed ID

- internal coherence,
- valid mutation,
- compatibility with the 36-bit model.

4. The 36-bit transmitted directional key

- fractal coherence,
- absence of drift,
- absence of falsification.

If a single bit is incoherent → session destroyed.

3.5. Model B — Validation of the Full Directional Key (128 bits)

This model is used when programmers choose **not** to transmit the key, but to reconstruct it entirely inside the volumetric private torus.

The Glass Fortress verifies:

1. the 20-bit starting point,
2. the 16-bit transmitted ID,
3. the 16-bit reconstructed ID,
4. the full 128-bit directional key.

The 128-bit directional key is:

- the only key that mutates,
- the only key that inverts,
- the only key that oscillates,
- the only key that moves inside the volume.

If the directional key is incoherent → session destroyed.

3.6. Structural Rule: Cell IDs May Repeat

✓ A cell ID may repeat without any limit. ✓ An ID may be shared by: → 10 cells, → 100 cells, → 1,000 cells, → 50,000 cells, → or more.

✓ This does **not** compromise security, because it relies on:

- volumetric directional geometry,
- internal mutations,
- 3D directional movements,
- fractal coherence,
- normal/ghost inversion,
- directional mutation by traversal.

✓ 50% of the 16 bits are reserved for directional mutations. ✓ The other 50% are active IDs, free and unconstrained.

✓ The 1,000,000 cells of the private torus may:

- massively share the same IDs,
- regroup into 3D directional clusters,
- mutate toward already-used IDs,
- overlap fractally.

This is **not** a collision → it is a normal, expected, validated directional property.

Role of the Glass Fortress

It validates that:

- ID repetitions are legal,
- mutations toward already-used IDs are coherent,
- internal fractal stability remains intact despite clustering,
- no directional drift is detected.

3.7. Validation of File Reconstruction

The Glass Fortress verifies:

- coherence of the directional key (36 or 128 bits),
- coherence of the volumetric torus,
- fractal coherence,
- absence of falsification,

- absence of drift.

It also verifies that:

- the file size is never transmitted,
- the file size is never visible,
- reconstruction is purely 3D directional.

If any incoherence is detected:

- the session is destroyed,
- the private torus is erased,
- the directional key is destroyed,
- reconstruction is cancelled,
- the IP is blocked.

3.8. Structural Protection

The Glass Fortress protects:

✓ the volumetric private torus ✓ the directional key of the active model ✓ the reconstructed file ✓ fractal coherence ✓ 3D directional coherence

It makes any falsification impossible by design.

3.9. Automatic Destruction

If an incoherence is detected:

- the session is destroyed,
- the private torus is erased,
- the directional key is destroyed,
- reconstruction is cancelled,
- the IP is blocked.

Destruction is:

- immediate,
- irreversible,

- total.

3.10. Summary of Chapter 3

The Glass Fortress:

- validates the directional key (36 or 128 bits),
- validates the transmitted ID,
- validates the reconstructed ID,
- validates the key jointly with the Host,
- validates file reconstruction,
- accepts repeated IDs without limit,
- protects the volumetric private torus,
- destroys the session upon anomaly,
- monitors volumetric directional mutation,
- guarantees 3D directional coherence.

It is the **structural directional firewall**, inviolable by design, at the core of **TorUsNet 9.0**.

Chapter 4 — Private Torus (Internal Structure, Geometry, Files, Directions, IDs) — TorUsNet 9.0

4.1. Overview

The Private Torus is the internal volumetric directional structure used to:

- traverse internal 3D directions,
- execute circular exponential jumps,
- mutate IDs along all three axes,
- invert normal/ghost bits,
- reconstruct the file,
- stabilize volumetric fractal coherence.

It contains:

- **1,000,000 cells**, organized into **100 canvases of 100×100**,
- each canvas represents a directional layer,
- the 100 canvases form a **100×100×100 volumetric torus**.

Each cell contains:

- a 16-bit ID,
- a normal bit,
- an implicit ghost bit,
- 4 internal directions,
- a 3D toric displacement,
- a circular exponential jump.

The Private Torus is programmed using **two distinct files**, read in parallel.

4.2. Internal Structure of the Volumetric Torus

Each cell of the torus contains:

✓ 1. A 16-bit ID

- 100% of possible values (0 to 65535) are used.
- IDs may repeat without any limit.
- No uniqueness, collision, or availability constraints.

✓ 2. A normal bit

- 0 = normal
- 1 = ghost (implicit, never transmitted)

✓ 3. Four internal directions

Encoded on 2 bits:

Code Direction

00 direction 1

01 direction 2

Code Direction

10 direction 3

11 direction 4

✓ 4. Volumetric fractal coherence

Each cell has internal coherence that:

- updates on every mutation,
- updates on every normal/ghost inversion,
- updates on every 3D directional displacement.

4.3. Internal Organization: 100 Canvases of 100×100 (100 Layers)

The Private Torus is structured into **100 independent canvases**, each 100×100 cells.

Each canvas represents:

- a directional layer,
- an autonomous internal surface,
- an independent fractal plane,
- a volumetric mutation level.

The 100 canvases are read:

- in parallel,
- in depth,
- as 100 stacked directional layers,
- forming a **100×100×100 volumetric torus**.

This structure:

- enables directional mutation by traversal (new in 9.0),
- enables cyclic intensity mod 100,
- enables mutation propagation across all three axes,
- makes internal geometry impossible to derive.

4.4. Cardinal, Diagonal, and Volumetric Movements (3D Toric Behavior)

Internal movements behave like a perfect torus across all three axes:

- exceeding upward → arrival at bottom
- exceeding downward → arrival at top
- exceeding left → arrival at right
- exceeding right → arrival at left
- exceeding depth → return to surface
- exceeding surface → return to depth

This behavior:

- removes borders,
- removes limits,
- removes dead zones,
- guarantees perfect continuity,
- makes it impossible to exit the volumetric torus.

Diagonal and volumetric movements follow the same logic.

4.5. Exponential Jumps (3D Circular Behavior)

Each internal direction contains a circular exponential jump.

This jump:

- does not follow cardinal axes,
- does not follow diagonals,
- does not follow the 3D grid,
- does not follow the geometry of the volumetric torus.

It follows an **internal circular orbit**, independent of 3D geometry.

✓ The exponential jump moves along a circle. ✓ When it reaches the end of the circle, it returns to the beginning. ✓ It can never leave the orbit. ✓ It can never get stuck. ✓ It can never drift.

This circular behavior:

- amplifies directional reach,
- creates an internal geometry independent of the 3D torus,
- makes paths impossible to reverse,
- stabilizes fractal coherence,
- prevents any external derivation.

4.6. Hybrid Geometry: 3D Torus + Circle

The internal structure combines two geometries:

✓ 3D Toric Geometry

For cardinal, diagonal, and volumetric movements.

✓ Circular Geometry

For exponential jumps.

This duality:

- creates a living directional surface,
- modifies geometry depending on movement type,
- makes paths impossible to reverse,
- stabilizes fractal coherence,
- prevents any external derivation.

4.7. Software Architecture of the Private Torus

The Private Torus is programmed using **two distinct files**, each with a precise role.

4.7.1. Normal Bits File (pure binary, loaded in RAM)

This file:

- contains only normal bits,
- is a pure binary file,
- is fully loaded into RAM,
- contains no directions,

- contains no IDs,
- contains no exponential jumps.

Loading into RAM enables:

- instant mutation of cells,
- ultra-fast reading of normal bits,
- continuous file reconstruction,
- immediate normal/ghost inversion.

4.7.2. Directional File (text, stored on disk)

This file:

- remains on disk,
- is a text file,
- contains the 4 directions of each cell,
- contains exponential jumps,
- contains the IDs of each cell,
- contains **1,000,000 lines**,
- is processed in parallel with the binary file.

Structure of a line:

Code

<ID> 101 2^5, 110 3^7, 101 2^7, 111 4^7

4.8. ID Mutation (TorUsNet 9.0 Update)

Two types of mutation exist, depending on the chosen directional model.

4.8.1. Local Mutation (36-bit model)

In the 36-bit model:

- the transmitted key provides only the starting cell and its ID,
- the key does not traverse the torus,
- local mutation is performed by the internal directional path,

- the key never mutates directly,
- no normal/ghost inversion is performed by the key.

Local mutation:

$$ID_{\text{cell}} \leftarrow ID_{\text{result of path}}$$

4.8.2. Structural Mutation (128-bit model)

In the 128-bit model:

- the internal key traverses the 3D torus,
- inverts normal/ghost bits,
- reconstructs the file,
- mutates the global torus ID,
- mutates the starting cell ID,
- modifies volumetric fractal coherence.

Structural mutation:

$$ID_{\text{torus}} \leftarrow ID_{\text{random}}$$

$$ID_{\text{start cell}} \leftarrow ID_{\text{random}}$$

4.9. Normal / Ghost

✓ The 128-bit model **can** invert normal/ghost. ✗ The 36-bit model **cannot** invert normal/ghost.

4.10. Torus ID

The torus has a **global 16-bit ID**.

This ID:

- changes with each internal key (128-bit model),
- is chosen randomly,
- may repeat freely,
- has no uniqueness constraint.

4.11. Role of the Torus in File Reconstruction

The Private Torus:

- reconstructs the file bit by bit,
- stabilizes volumetric fractal coherence,
- applies directional mutations,
- applies normal/ghost inversions,
- applies exponential jumps,
- applies 3D toric movements.

The file size:

- is never transmitted,
- is never visible,
- is deduced directionally.

4.12. Summary of Chapter 4

The Private Torus:

- contains 100 canvases of 100×100, read as 100 directional layers,
- forms a 100×100×100 volumetric torus,
- contains 1,000,000 cells,
- each cell has a 16-bit ID,
- IDs may repeat freely and massively,
- cardinal movements are 3D toric,
- exponential jumps are circular,
- hybrid geometry makes paths impossible to reverse,
- the torus is programmed with two files (RAM binary + disk directional),
- the 36-bit model provides the starting cell and its ID,
- local mutation (36-bit) is performed by the internal path,
- the 128-bit model mutates structurally,
- normal/ghost inversion depends on the chosen model,

- fractal coherence is continuously updated,
- volumetric directional mutation by traversal is applied (new in 9.0),

The hybrid structure makes the torus **inviolable by design**.

Chapter 5 — ID System (16 bits, repetition, mutation, role)

TorUsNet 9.0 — complete update

5.1. Overview

The directional protocol uses an identification system entirely based on:

- 16-bit IDs,
- 100% of all possible values,
- IDs repeatable without any limit,
- random mutations,
- local or structural redefinitions,
- interaction depending on the chosen directional model:
 - **36-bit model** (local mutation via volumetric traversal),
 - **128-bit model** (structural mutation).

IDs are at the core of:

- the volumetric Private Torus,
- the 1,000,000 cells,
- 3D directional mutations,
- fractal coherence,
- file reconstruction.

5.2. ID Size: 16 bits

All protocol IDs are 16-bit:

- cell ID,
- torus ID,

- transmitted ID (36-bit model),
- reconstructed ID (128-bit model),
- locally mutated ID (36-bit model),
- structurally mutated ID (128-bit model).

✓ Possible values

0 → 65535

✓ 100% of values are used

No value is:

- reserved,
- forbidden,
- set aside.

5.3. ID Repetition

IDs may repeat freely, without any limit.

Multiple cells may share the same ID, including:

- tens,
- hundreds,
- thousands,
- or tens of thousands of cells.

✓ Structural reason

- The ID is **not** a unique identifier.
- The ID is a **directional marker**.
- Security relies on **3D directional geometry**, not on ID uniqueness.

✓ Consequence

- 1,000,000 cells,
- 65,536 possible IDs,
- normal repetitions,

- normal collisions,
- no impact on volumetric fractal coherence.

5.4. ID Mutation

Two types of mutation exist, depending on the chosen directional model.

5.4.1. Local Mutation (36-bit model)

(TorUsNet 9.0 update)

In the 36-bit model:

- the key provides only:
 - the starting cell,
 - the starting ID;
- the key does **not** traverse the torus,
- the key performs **no movement**,
- local mutation is performed by the **volumetric directional path**,
- no other cell is directly modified,
- the key performs **no normal/ghost inversion**.

Local mutation:

$$ID_{\text{cell}} \leftarrow ID_{\text{result of path}}$$

Mutation is therefore a **consequence of 3D directional movement**, not a direct action of the key.

5.4.2. Structural Mutation (128-bit model)

In the 128-bit model:

- the internal key traverses the volumetric torus,
- inverts normal/ghost bits,
- reconstructs the file,
- mutates the global torus ID,
- mutates the starting cell ID,

- modifies 3D fractal coherence.

Structural mutation:

$$ID_{\text{torus}} \leftarrow ID_{\text{random}}$$

$$ID_{\text{start cell}} \leftarrow ID_{\text{random}}$$

Random IDs are chosen on 16 bits, even if already used.

5.5. Random Mutation on 16 Bits

During mutation, the program chooses a random 16-bit ID, even if that ID is already used.

✓ Properties

- mutation = random draw from 0–65535,
- no uniqueness constraint,
- no collision constraint,
- no availability constraint,
- mutation is purely directional.

5.6. Role of IDs in the Volumetric Private Torus

Each cell of the torus contains:

- a 16-bit ID,
- a normal bit,
- an implicit ghost bit,
- four internal directions,
- a 3D toric displacement,
- a circular exponential jump.

The ID:

- identifies the cell in the directional file,
- serves as an entry point for mutations,
- serves as a marker for fractal coherence,
- serves as a reference for internal reconstruction.

5.7. Torus ID

The torus has a **global 16-bit ID**.

This ID:

- changes with each internal key (128-bit model),
- is chosen randomly,
- may repeat freely,
- has no uniqueness constraint.

The 36-bit model **never** modifies the global torus ID.

5.8. Interaction Between IDs and Directional Models

✓ 36-bit model

- provides the starting cell,
- provides the starting ID,
- triggers an internal 3D traversal,
- local mutation is performed by the path,
- never modifies the torus ID,
- never inverts normal/ghost.

✓ 128-bit model

- traverses the volumetric torus,
- inverts normal/ghost,
- reconstructs the file,
- mutates the torus ID,
- mutates the starting cell ID,
- stabilizes 3D fractal coherence.

5.9. Fractal Coherence and IDs

Fractal coherence:

- accepts repeated IDs,

- accepts collisions,
- accepts random mutations,
- stabilizes automatically,
- updates on every 3D directional mutation.

IDs are **not** a security structure. They are a **directional structure**.

5.10. Summary of Chapter 5

The ID system works as follows:

- all IDs are 16-bit,
- 100% of values are used,
- IDs may repeat freely,
- mutations choose a random ID,
- the 36-bit model provides the starting cell and mutates via internal 3D traversal,
- the 128-bit model mutates the torus and the starting cell,
- ID collisions are normal,
- fractal coherence remains stable,
- security relies on **volumetric directional geometry**, not on ID uniqueness.

Chapter 6 — Directional Handshake (Host ↔ Client Sequence)

TorUsNet 9.0 — complete update

6.1. Overview

The directional handshake is the structural mechanism that establishes a coherent session between a Host and a Client.

Unlike classical protocols (TLS, SSH, IPSec) which rely on:

- private keys,
- certificates,
- signatures,

- mathematical derivations,
- cryptographic exchanges,

the directional handshake relies exclusively on:

- the internal geometry of the $100 \times 100 \times 100$ tori,
- volumetric fractal coherence,
- 3D directional movements,
- internal IDs,
- joint validation by the Host + Glass Fortress.

It contains **no cryptography, no derivation, no exploitable mathematical structure.**

6.2. Objectives of the Handshake

The directional handshake ensures:

1. validation of the first directional key (36 bits or 128 bits),
2. initial synchronization between Host and Client,
3. registration of the Client's IP as a trusted IP,
4. creation of the global session ID (16 bits),
5. activation of the volumetric Glass Fortress,
6. initialization of the Private Torus (100 canvases of 100×100),
7. establishment of the global torus ID (16 bits),
8. complete directional synchronization through **5 transmissions.**

These steps are **structural**, not cryptographic.

6.3. The Official Sequence: 5 Transmissions

The directional sequence consists of **5 transmissions**, each with a precise structural role. It is mandatory, inviolable, and forms the foundation of directional synchronization.

Transmission 1 — Host → Client

Initial sending of the directional key

The Host sends a directional key:

- **36-bit model:**
 - 20 bits: starting cell
 - 16 bits: starting ID
- **128-bit model:**
 - 128 bits: full directional key

The Client:

- receives the key,
- verifies its structure,
- verifies its model (36 or 128 bits),
- verifies the targeted ID,
- verifies initial fractal coherence,
- sends the key to the Glass Fortress for joint validation.

No directional execution occurs yet.

Transmission 2 — Client → Host

Returning the key to confirm reception

The Client sends back **the exact same key** to the Host.

The Host:

- compares the sent key and the received key,
- validates reception,
- registers the Client's IP as a trusted IP,
- activates the volumetric Glass Fortress.

If the key differs → **immediate destruction**.

Transmission 3 — Host → Client

Authorization to execute

The Host sends a second copy of the key.

The Client:

- compares the returned key with the previously received one,
- validates coherence,
- executes the internal directional path inside its Private Torus (100 layers),
- updates internal paths,
- updates volumetric fractal coherence.

Effects depending on the model:

✓ 36-bit model

- local mutation of the starting cell's ID,
- mutation performed by the 3D directional path,
- no normal/ghost inversion.

✓ 128-bit model

- normal/ghost inversion,
- structural mutation of the torus ID,
- structural mutation of the starting cell ID,
- volumetric directional mutation (100 mutated layers if 100 traversals occur).

The Client's Private Torus becomes:

- living,
- oscillating,
- modified.

Transmission 4 — Client → Host

Confirmation of path execution

The Client returns the key **after executing the internal path**.

This informs the Host that:

- local mutation (36 bits) or structural mutation (128 bits) has been performed,
- internal paths have been updated,
- fractal progression is coherent,

- the Client's Private Torus is up to date.

The Host can then:

- execute the same path,
- apply the same mutations,
- synchronize its Private Torus with the Client's.

Transmission 5 — Host → Client

Final synchronization confirmation

The Host sends the key one last time to confirm:

- it has executed the path,
- it has applied the mutations,
- its Private Torus is synchronized with the Client's,
- the session is directionally stable.

The Client receives this confirmation and the session becomes:

- coherent,
- stable,
- inviolable.

6.4. Why Five Transmissions?

6.4.1. Complete Bidirectional Validation

Host validates Client. Client validates Host. Then each validates the other's directional execution.

6.4.2. Protection Against Injections

An attacker cannot:

- intercept the key,
- modify it,
- resend it,
- falsify it.

Any incoherence is detected structurally.

6.4.3. Protection Against Repetition

A repeated key is:

- detected,
- rejected,
- destroys the session.

6.4.4. Protection Against Derivation

An attacker cannot derive:

- the starting cell,
- the starting ID,
- internal 3D movements,
- mutations,
- fractal coherence.

The directional structure is **non-derivable**.

6.4.5. Perfect Directional Synchronization

The 5 transmissions ensure that:

- both private tori executed the same path,
- mutations are identical,
- internal geometry is coherent,
- no divergence is possible.

6.5. Initial Synchronization

Once the key is validated:

- the Authentication Torus is used only once,
- the Private Torus (100 canvases) is initialized,
- the global ID is created,
- normal/ghost pairs are defined (128-bit model),

- internal paths are activated,
- the global torus ID is established,
- the Glass Fortress is armed.

This synchronization is **fractal**, not mathematical.

6.6. Initial Reconstruction of the Private Torus

The first key triggers:

- directional initialization,
- activation of internal paths,
- local mutation (36 bits) or structural mutation (128 bits),
- stabilization of volumetric fractal coherence,
- activation of internal directions,
- activation of exponential jumps.

The Private Torus becomes:

- living,
- oscillating,
- operational.

6.7. Error Handling During the Handshake

6.7.1. Incoherent Key

If the key:

- contains an invalid ID,
- contains an impossible cell,
- contains an illegitimate model,

→ the session is destroyed.

6.7.2. Modified Key

If the returned key differs:

- immediate destruction,

- erasure of the Private Torus,
- IP invalidation.

6.7.3. No Response

If the Host receives no response after 5 seconds:

- the key is considered lost,
- a new key is generated,
- with a different starting cell,
- but reconstructing the same directional structure.

No retransmission of the same packet is ever performed.

6.8. Summary of Chapter 6

The directional handshake:

- relies on 5 transmissions,
- validates the first directional key,
- registers the Client's IP as trusted,
- initializes the Private Torus (100 canvases),
- activates the volumetric Glass Fortress,
- creates the global ID,
- establishes the unique torus ID,
- guarantees total directional coherence,
- protects against injections, repetitions, derivations, and falsifications.

It forms the foundation of **structural directional security**.

Chapter 7 — Multi-Torus and Parallel Networks

TorUsNet 9.0 — complete update

7.1. Overview

The multi-torus system is one of the most powerful foundations of the TorUsNet 9.0 directional architecture. It enables the creation of multiple parallel directional networks, each possessing:

- its own 100×100×100 Private Torus,
- its own Authentication Torus,
- its own directional keys (36 bits or 128 bits),
- its own internal IDs (16 bits),
- its own global session ID,
- its own global torus ID,
- its own volumetric Glass Fortress.

The **Public Torus**, pre-generated and universal, serves as the common directional base for all branches.

Each parallel network is completely isolated from the others, even if they coexist:

- on the same machine,
- on the same system,
- in the same network environment.

The multi-torus system enables:

- directional segmentation,
- structural sovereignty,
- total branch isolation,
- creation of inviolable private networks,
- controlled directional duplication.

7.2. Definition of a Parallel Network

A parallel network is a complete instance of the directional architecture, consisting of:

1. The Public Torus (pre-generated, universal)
2. An Authentication Torus
3. A 100×100×100 Private Torus

4. A set of directional keys
5. A volumetric Glass Fortress
6. A global session ID
7. A global torus ID

Each parallel network is an independent directional universe.

It shares:

- no keys,
- no IDs,
- no private torus,
- no fractal coherence,
- no internal paths.

7.3. Creation of a New Directional Branch

Creating a new branch follows three steps.

7.3.1. Use of the Pre-Generated Public Torus

The Public Torus:

- is pre-generated,
- is provided with the software,
- is universal,
- is identical for all installations,
- is non-modifiable,
- is non-derivable.

It provides the common directional base but contains no private information.

7.3.2. Generation of the Authentication Torus

An Authentication Torus is generated. It validates the first directional key and initializes:

- the first normal/ghost pairs (128-bit model),
- the first internal paths,

- the global session ID,
- the global torus ID.

This torus is used **only once**: during the initial handshake.

7.3.3. Generation of the Volumetric Private Torus

A $100 \times 100 \times 100$ Private Torus is generated. It will be fractally reconstructed from directional keys:

- normal/ghost inversion (128-bit model),
- ID mutation (local or structural),
- activation of internal paths,
- volumetric fractal stabilization,
- directional mutation by traversal (100 mutated layers if 100 traversals occur).

Each Private Torus is independent from all others.

7.4. Directional Branches

Each parallel network is called a **directional branch**.

A branch possesses:

- its own $100 \times 100 \times 100$ Private Torus,
- its own fractal progression,
- its own internal IDs,
- its own Glass Fortress,
- its own global session ID,
- its own global torus ID.

7.4.1. Independent Branches

Branches do not communicate with each other. They cannot:

- share keys,
- share IDs,
- share private tori,

- share internal paths,
- share mutations.

7.4.2. Sovereign Branches

Each branch is sovereign:

- it does not depend on any other branch,
- it cannot be influenced by another branch,
- it cannot be derived by another branch.

7.5. Directional Duplication (Offline Primary Torus)

A key feature of the multi-torus system is **offline directional duplication**.

7.5.1. Primary Torus

A primary torus may be:

- generated,
- stored offline,
- transferred physically,
- used to create a new branch.

7.5.2. Offline Transfer

The primary torus may be transferred:

- on a USB drive,
- on a hard disk,
- on any physical medium,
- on any non-network medium.

This transfer contains **no directional keys**. It contains only:

- the 100×100×100 directional geometry,
- internal IDs,
- internal directions,
- exponential jumps,

- the 100 canvases of 100×100.

7.5.3. Creating a Branch from the Primary Torus

Once the primary torus is transferred:

- a new branch is created,
- a new volumetric Private Torus is reconstructed,
- a new global session ID is generated,
- a new global torus ID is established,
- a new Glass Fortress is activated.

The new branch is completely independent from the original branch.

7.6. Structural Isolation

Structural isolation is total.

7.6.1. Fractal Isolation

Fractal actions of one branch:

- cannot influence another branch,
- cannot derive another branch,
- cannot reconstruct another branch.

7.6.2. Directional Isolation

Internal paths of one branch:

- cannot be used in another branch,
- cannot be derived by another branch.

7.6.3. ID Isolation

Internal IDs:

- are not compatible between branches,
- cannot be transferred,
- cannot be derived.

7.7. Multi-Torus and Post-Quantum Security

The multi-torus system strengthens post-quantum security:

A quantum computer cannot derive:

- a Private Torus,
- a branch,
- a directional key,
- an ID,
- the global torus ID.

Even with infinite power, an attacker cannot:

- explore a Private Torus,
- derive a Private Torus,
- reconstruct a Private Torus,
- analyze a Private Torus.

Private tori are **geometrically inviolable**.

7.8. Multi-Torus and Photonic Security

With the arrival of photonic computers:

- mathematical structures become obsolete,
- classical cryptography becomes vulnerable.

The multi-torus system is designed for this era:

Private tori are:

- non-transmissible,
- non-derivable,
- non-analyzable,
- non-explorable.

A photonic computer cannot:

- penetrate a Private Torus,
- analyze a Private Torus,

- derive a Private Torus.

7.9. Summary of Chapter 7

The multi-torus system enables:

- creation of parallel networks,
- total isolation of branches,
- offline directional duplication,
- structural sovereignty,
- post-quantum security,
- photonic security.

Each branch is an independent directional universe, inviolable by design. The pre-generated universal Public Torus serves as the common directional base — but **never** as a shared Private Torus.

Chapter 8 — Directional Ports and Internal Communication Geometry

TorUsNet 9.0 — complete update

8.1. Overview

The directional ports of TorUsNet 9.0 are a structural extension of classical ports. Unlike TCP/UDP ports (0–65535), TorUsNet uses:

- extended ports (several billions of possibilities),
- mutable ports,
- non-scannable ports,
- non-derivable ports,
- non-falsifiable ports,
- ports structurally validated by the volumetric Glass Fortress.

A directional port is **not**:

- an integer,
- a socket,

- a TCP channel,
- a network identifier.

A directional port is an **internal structure** of the 100×100×100 Private Torus, generated directionally, mutated at each torus renewal, and mutated at each new connection.

8.2. Definition of a Directional Port

A directional port is:

- an extended port number (billions of possibilities),
- generated by the volumetric Private Torus,
- mutated at each torus renewal,
- mutated at each new connection,
- impossible to predict,
- impossible to scan,
- impossible to derive.

Structural rule: one connection per port

Each directional port:

- allows only one connection,
- is invalidated as soon as the session ends,
- is replaced by a new extended port at the next connection.

8.3. Why Directional Ports Are Inviolable

8.3.1. Absence from the Classical Network Layer

An attacker cannot:

- scan ports,
- probe ports,
- open a port,
- send a packet to a port.

Directional ports **do not exist** in the classical network layer. They exist only inside the internal mechanics of the 100×100×100 Private Torus.

8.3.2. Mutable and Non-Derivable Ports

Directional ports:

- mutate automatically,
- are never reused,
- cannot be predicted,
- cannot be derived mathematically.

8.3.3. Validation by the Glass Fortress

The volumetric Glass Fortress:

- monitors directional ports,
- validates their fractal coherence,
- destroys the session if a port is violated,
- blocks the hostile IP in the operating system firewall.

8.3.4. Non-Transmissible Ports

A directional port:

- cannot be sent,
- cannot be copied,
- cannot be exported,
- cannot be derived.

It exists **only locally**, inside the Private Torus.

8.4. Mutation of Directional Ports

8.4.1. Mutation at Torus Renewal

Each time the Private Torus is regenerated:

- a new set of extended ports is generated,
- all previous ports become invalid.

8.4.2. Mutation at Each New Connection

At each new session:

- a new extended port is generated,
- only one connection is allowed on this port,
- any second connection attempt triggers immediate destruction.

8.5. Directional Ports and Transmission

Host → Client

The Host opens an extended directional port. The Client must respond on the **same port**.

Client → Host

The Client returns the directional key on this port. The Host validates coherence.

Host → Client

The Host returns the final key on the same port.

Port Violation

If a directional port is violated:

- the session is destroyed,
- the Private Torus is erased,
- the IP is blocked.

8.6. Directional Ports and Multi-Torus

In a multi-torus environment:

- each branch has its own directional ports,
- ports are not compatible between branches,
- ports cannot be derived from one branch to another.

A directional port is strictly local to its directional branch.

8.7. Directional Ports and Post-Quantum Security

Directional ports are inviolable by a quantum computer:

- impossible to derive an extended port,
- impossible to predict a mutation,
- impossible to explore the 100×100×100 Private Torus,

- impossible to reconstruct internal ports.

8.8. Directional Ports and Photonic Security

Photonic computers can break:

- RSA,
- ECC,
- TLS,
- SSH.

But they cannot:

- analyze volumetric directional geometry,
- derive a torus,
- reconstruct a directional port,
- predict a mutated extended port.

Directional ports are **structurally inviolable**.

8.9. Summary of Chapter 8

Directional ports:

- are classical ports extended to billions of possibilities,
- mutate at each torus renewal,
- mutate at each new connection,
- allow only one connection per port,
- are non-scannable,
- non-derivable,
- non-falsifiable,
- validated by the Glass Fortress,
- inviolable by design.

They form the **internal communication layer** of TorUsNet 9.0, completely independent from classical network protocols.

Chapter 9 — DNS and HTTPS Authentication (Temporary Trust Layer)

TorUsNet 9.0 — complete update

9.1. Overview

Before the 100×100×100 directional geometry takes full control of the session, a temporary trust layer is required to:

- identify the requested Host,
- avoid malicious redirections,
- prevent classical MITM attacks,
- secure the transmission of the first directional key,
- guarantee that the Client is contacting the correct Host.

This layer relies on two classical Internet technologies:

- **DNS** (domain name resolution),
- **HTTPS** (TLS certificate).

These technologies are **not** part of the directional mechanics. They serve only as an entry buffer before activation of:

- the Authentication Torus,
- the Private Torus (100 canvases of 100×100),
- the volumetric Glass Fortress,
- internal IDs,
- the global ID,
- the global torus ID.

Once the first key is validated, DNS and HTTPS disappear completely.

9.2. Role of DNS in TorUsNet 9.0

DNS is used for **one single purpose**:

Identify the requested Host before entering directional geometry.

9.2.1. Domain Verification

When a Client contacts a Host:

1. DNS resolves the domain name.
2. The protocol verifies that this domain belongs to the list of trusted Hosts.
3. If the domain is unknown or suspicious:
 - the connection switches to user-confirmation mode,
 - no directional key is transmitted.

9.2.2. Protection Against Redirections

DNS protects against:

- malicious redirections,
- fake domains,
- cloned Hosts,
- traffic hijacking.

9.2.3. Strictly Temporary Role

Once the first key is validated:

DNS is no longer used. Directional geometry takes over.

9.3. Role of HTTPS in TorUsNet 9.0

HTTPS is used to:

Secure the transmission of the first directional key.

9.3.1. Certificate Verification

The Client verifies:

- the TLS certificate,
- the validity of the domain,
- the absence of classical MITM interception.

If the site does not support HTTPS:

- the connection switches to user-confirmation mode,
- no directional key is sent.

9.3.2. Protection Against Classical Attacks

HTTPS protects against:

- interceptions,
- injections,
- traffic manipulation,
- downgrade attacks.

9.3.3. Strictly Temporary Role

Once the first key is transmitted:

HTTPS disappears. The session becomes **100% directional**.

9.4. Interaction DNS ↔ HTTPS ↔ Authentication Torus

The complete sequence is:

1. DNS identifies the Host.
2. HTTPS secures the first transmission.
3. The Authentication Torus validates the first key.
4. The Glass Fortress activates the directional session.
5. The Private Torus (100×100×100) takes control.
6. DNS and HTTPS are abandoned.

9.4.1. Full Trust Case

- DNS valid
- HTTPS valid

→ The first key is transmitted automatically.

9.4.2. Partial Trust Case

- DNS valid
- HTTPS absent

→ User confirmation required.

9.4.3. Suspicion Case

- DNS suspicious

- HTTPS absent or invalid

→ Connection blocked until explicit validation.

9.5. Why TorUsNet Still Uses DNS and HTTPS

TorUsNet is a **post-Internet directional architecture**. However, today's ecosystem still requires:

- domain names,
- TLS certificates,
- HTTP(S) servers.

DNS and HTTPS therefore act as a bridge between:

- classical Internet,
- directional Internet.

9.5.1. Final Objective: Complete Removal

In future versions:

- DNS will be replaced by **directional DNS**,
- HTTPS will be replaced by **native directional validation**,
- certificates will be replaced by **directional IDs**,
- connections will be entirely geometric.

9.6. Interaction with Directional Updates (TorUsNet 9.0)

ID Inversion and Mutation

Once the first key is validated:

- the starting cell's ID mutates (36-bit or 128-bit model),
- the global ID is created,
- the global torus ID is established,
- DNS and HTTPS are no longer needed.

Autonomous Directional Session

The session becomes:

- continuous,
- living,
- autonomous,
- fractal,
- volumetric (100 layers),
- non-transmissible,
- non-derivable.

9.7. Summary of Chapter 9

This chapter establishes that:

- DNS and HTTPS serve only to secure the first key,
- they disappear immediately after validation,
- the session becomes 100% directional,
- the Glass Fortress takes control,
- the Private Torus (100×100×100) becomes the sole communication layer,
- internal IDs replace certificates,
- directional ports replace sockets,
- directional geometry replaces cryptography.

Chapter 10 — Logic for Reading a Cell, the Layers, and the Intensity

TorUsNet 9.0 — complete update

10.1. Overview

Reading a cell in TorUsNet 9.0 relies on three fundamental directional mechanisms:

1. **Circular reading** triggered by an exponential jump followed by an operator (+, −, ×, ÷)
2. **Layer reading** (100 canvases of 100×100) treated as a vertical torus
3. **Intensity logic (mod 100)** determining the layer being read and the layers being mutated

These three mechanisms merge to create a reading process that is:

- volumetric,
- circular,
- non-linear,
- non-invertible,
- impossible to derive,
- living and fractal.

10.2. Logic of Layer Reading (100 canvases = 100 layers)

The Private Torus of TorUsNet 9.0 is composed of:

100 canvases of 100×100 cells, i.e., 100 vertical layers, each containing 10,000 cells.

10.2.1. Directional Vertical Reading

When the directional path requests the reading of a layer:

- the layer is read line by line,
- from cell 1 to cell 10,000,
- **as a circle** (see section 10.3).

10.2.2. Vertical Overflow

If reading goes beyond layer 100:

- it returns to layer 1.

If reading goes below layer 1:

- it returns to layer 100.

10.2.3. Volumetric Reading

Reading may traverse:

- a single layer,
- multiple layers,
- all layers,

depending on the intensity and the exponential jump.

10.3. Logic of Circular Reading (exponential jump + operator)

When the directional path performs:

an exponential jump followed by an operator (+, −, ×, ÷)

the reading switches into **circular mode**.

10.3.1. Circular Reading of a Canvas

- The 10,000 cells are read as a circle.
- Cell 10,000 loops back to cell 1.
- No edge exists.
- Reading can never stop on a boundary.

10.3.2. Circular Reading of Layers

- Layer 100 loops back to layer 1.
- Layer 1 loops back to layer 100.
- The 100 layers form a **vertical ring**.

10.3.3. Effect of Operators

Operator Directional Effect

+	accelerated circular forward motion
−	reversed circular backward motion
×	amplification of reading speed
÷	fractal reduction of reading speed

10.4. Logic of Intensity (mod 100)

Intensity is a directional number determining:

- the layer being read,
- the number of layers traversed,
- the number of layers mutated,
- the directional depth of the path.

10.4.1. Calculation of the Layer Read

layer read = intensity 100

Examples:

- Intensity 254 $\rightarrow 254 \bmod 100 = \mathbf{54} \rightarrow$ read layer 54
- Intensity 999 $\rightarrow 999 \bmod 100 = \mathbf{99}$
- Intensity 10,000 $\rightarrow 10,000 \bmod 100 = \mathbf{0} \rightarrow$ layer **100**

10.4.2. Traversal of Layers

If intensity exceeds 100:

- it loops,
- returns to layer 1,
- continues its path.

10.4.3. Mutation of Traversed Layers

Directional rule (TorUsNet 9.0):

If intensity traverses N layers, then those N layers mutate.

Example:

- Intensity = 254
- $254 \bmod 100 = 54$
- Traversal: $100 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 54$
- Number of layers traversed: **54**
- **All 54 layers mutate.**

10.5. Automatic Bit Inversion When Reading a Layer

TorUsNet 9.0 introduces a fundamental directional rule:

When a layer is read, all traversed bits invert automatically.

This inversion applies to:

- the normal bit,
- the implicit ghost bit,

- every cell traversed,
- in every layer crossed,
- according to intensity logic (mod 100).

10.5.1. Inversion in the Layer Read

When intensity selects a layer (e.g., layer 54):

- circular reading traverses all 10,000 cells,
- each traversed cell inverts its normal/ghost bit,
- inversion is immediate,
- inversion is irreversible,
- inversion modifies the fractal coherence of the layer.

10.5.2. Inversion in Traversed Layers

If intensity traverses N layers:

- all N layers are read,
- all N layers mutate,
- **all bits traversed in those N layers invert,**
- even if the layer is not the final layer being read.

10.5.3. Interaction with Directional Models

36-bit model

- The key does **not** traverse the torus.
- The internal path triggers circular reading.
- Bits invert even though the 36-bit model does not invert normal/ghost by itself.
- **Inversion is caused by reading, not by the key.**

128-bit model

- The key **does** traverse the torus.
- The 128-bit model already inverts normal/ghost.
- Reading adds an additional inversion, independent of the key.

- **Double inversion is possible** depending on the path.

10.6. Fusion of the Three Logics

When an exponential jump followed by an operator is executed inside a layer determined by intensity, reading becomes:

- circular within the layer,
- circular across all 100 layers,
- volumetric,
- mutating,
- fractal,
- non-invertible.

10.7. Complete Example

Intensity: **254** Exponential jump: 2^7 Operator: $\times$

Step 1 — Determine the Layer

$254 \bmod 100 = 54$

Step 2 — Traverse Layers

Traversal: $100 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 54 \rightarrow 54$ layers read $\rightarrow 54$ layers invert all their traversed bits

Step 3 — Circular Reading

Circular reading in **each** traversed layer.

Step 4 — Directional Mutation

Depending on the model:

- **36-bit model:** local mutation
- **128-bit model:** structural mutation

10.8. Summary of Chapter 10

Reading a cell in TorUsNet 9.0 relies on:

- circular reading triggered by exponential jump + operator,
- reading the 100 layers as a vertical ring,

- intensity mod 100 determining the layer read,
- mutation of traversed layers,
- automatic inversion of traversed bits,
- volumetric fractal coherence,
- fusion of circular + volumetric reading.

Directional reading becomes:

- living,
- autonomous,
- non-linear,
- non-invertible,
- impossible to derive,
- impossible to falsify.

Chapter 11 — New Logic for Layer Reading by Intensity

TorUsNet 9.0 — complete update

11.1. Overview

TorUsNet 9.0 introduces a new directional mechanic: the fusion between the **intensity transmitted over the network** and the **internal private intensity**.

Every directional movement inside the Private Torus now depends on **two intensities**:

- **Transmitted intensity** – included in the path sent over the network – visible to both Host and Client – fixed for the duration of the transmitted path
- **Private intensity** – internal to the Private Torus – never transmitted – never visible – changes at every movement – depends on the cell traversed – contains an internal additive or subtraction

Layer reading therefore becomes a **hybrid mechanism**, combining:

- external intensity (network),
- internal intensity (private),

- circular reading,
- volumetric reading,
- inversion of traversed bits,
- directional mutation.

11.2. Transmitted Intensity (external)

The transmitted intensity is included in the path sent over the network.

It is used to:

- determine the initial layer to read,
- determine the number of layers to traverse,
- synchronize Host ↔ Client,
- stabilize directional coherence between both toruses.

11.2.1. Calculation of the initial layer

$$\text{initial layer} = \text{transmitted intensity} \bmod 100$$

Examples:

- Transmitted intensity 87 → layer 87
- Transmitted intensity 254 → $254 \bmod 100 = 54$
- Transmitted intensity 10,000 → $10,000 \bmod 100 = 0 \rightarrow$ layer **100**

11.3. Private Intensity (internal)

Private intensity belongs exclusively to the Private Torus.

It:

- is never transmitted,
- is never visible,
- is never exported,
- changes at every movement,
- depends on the cell traversed.

Each cell of the torus contains:

- a 16-bit ID,
- a normal/ghost bit,
- 4 internal directions,
- an exponential jump,
- an internal additive or subtraction for private intensity.

11.3.1. Updating private intensity

At each movement:

$$I_{\text{private}} \leftarrow I_{\text{private}} + A_{\text{cell}}$$

Where:

- A_{cell} = internal additive or subtraction of the cell
- typical value: **-7 to +7**
- never transmitted
- never visible

11.3.2. Private intensity = living intensity

Private intensity:

- evolves at every movement,
- depends on internal geometry,
- depends on mutations,
- depends on normal/ghost inversions,
- depends on layers traversed.

It is **living, oscillating, non-derivable**.

11.4. Fusion of Intensities (external + internal)

Layer reading depends on:

$$I_{\text{final}} = I_{\text{transmitted}} + I_{\text{private}}$$

This final intensity determines:

- the layer read,

- the number of layers traversed,
- the number of layers mutated,
- the number of layers inverted,
- directional depth,
- circular reading speed.

11.4.1. Calculation of the final layer

$$\text{final layer} = I_{\text{final}} 100$$

11.5. Layer Reading by Final Intensity

Once the final intensity is calculated:

- The final layer is determined.
- The torus reads this layer line by line as a circle.
- Traversed layers are read as a vertical ring.
- All traversed bits invert.
- Traversed layers mutate.
- Fractal coherence is updated.

11.5.1. Traversal of layers

If $I_{\text{final}} = 254$:

- $254 \bmod 100 = \mathbf{54}$
- Traversal: $100 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 54$
- **54 layers read**
- **54 layers invert their bits**
- **54 layers mutate**

11.6. Interaction with Directional Models

36-bit model

- the key does not traverse the torus
- transmitted intensity is used

- private intensity evolves at each internal movement
- local mutation depends on the reading result
- traversed bits invert

128-bit model

- the key traverses the torus
- private intensity evolves at each movement of the key
- final intensity modifies circular reading
- normal/ghost inversion + reading inversion
- structural mutation depends on final intensity

11.7. Complete Example

Transmitted intensity: **200** Initial private intensity: **17** Traversed cell: internal additive = **+3**

Step 1 — Update private intensity

$$I_{\text{private}} = 17 + 3 = 20$$

Step 2 — Calculate final intensity

$$I_{\text{final}} = 200 + 20 = 220$$

Step 3 — Determine final layer

$$220 \text{ } 100 = 20$$

→ Read layer **20**

Step 4 — Traverse layers

Traversal: 100 → 1 → 2 → ... → 20 → **20 layers read** → **20 layers invert their bits** → **20 layers mutate**

Step 5 — Circular reading

Read line by line as a circle.

Step 6 — Directional mutation

Depending on the model:

- local mutation (36-bit)

- structural mutation (128-bit)

11.8. Summary of Chapter 11

The new logic for layer reading by intensity relies on:

- transmitted network intensity,
- internal private intensity,
- fusion of both intensities,
- circular reading,
- volumetric reading,
- automatic inversion of traversed bits,
- mutation of traversed layers,
- living fractal coherence.

This mechanic makes:

- every reading unique,
- every path non-reproducible,
- every torus impossible to derive,
- every session inviolable by design.