STARLIGHT

A Temporal Traversal-Based Encoding Protocol

Protocol Design Specification

Harsha C

May 20, 2026

Abstract

STARLIGHT is a human-operable encoding protocol designed to produce cognitively resistant text through temporal state mutation and traversal-based symbol reordering. Unlike traditional substitution ciphers, STARLIGHT achieves encoding variability through three mechanisms: coordinate-transfer mapping using weekday-dependent grid transformations, position-dependent cyclic shifts, and lexicographic word reordering based on encoded symbols. The protocol prioritizes pen-and-paper usability, deterministic reversibility, and psychological parsing disruption over computational security. This specification defines STARLIGHT v1, including the 7x7 symbol architecture, 7-day temporal cycle, complete encoding/decoding procedures, and application guidelines.

Contents

1 Introduction

STARLIGHT is a temporal state-dependent encoding protocol designed to challenge automatic human parsing behavior.

Unlike traditional substitution ciphers, STARLIGHT does not rely on a fixed alphabet mapping. Instead, the protocol evolves through time-dependent grid transformation and traversal-based reordering.

The system is designed for:

- Human pen-and-paper usability
- Deterministic reversibility
- Psychological parsing disruption
- Temporal mutation
- Stateful interpretation
- ARG and puzzle-system applications

STARLIGHT is not intended to replace natural language. Instead, it acts as a dynamic transformation layer applied to plaintext.

2 Core Philosophy

The foundational principle of STARLIGHT is:

> Meaning is not fixed. Meaning emerges from state, time, traversal, and transformation.

The protocol intentionally challenges assumptions humans make while reading:

- Left-to-right parsing
- Stable symbol meaning
- Static frequency distribution
- Fixed traversal order
- Immediate semantic reconstruction

The goal is not merely encryption.

The goal is controlled interpretive instability.

2.1 Language Assumptions Being Broken

Assumption	Normal Language	STARLIGHT Strategy
Stable Symbol Meaning	Letters retain fixed meaning	Mappings evolve through temporal state
Linear Parsing	Left-to-right reading	Traversal determined dynamically
Static Frequency	Predictable repetition	Mutation disrupts frequency analysis
Stable Context	Time-independent interpretation	Decoding depends on temporal state
Visible Structure	Order reflects meaning	Traversal order differs from visual order
Reliable Pattern Recovery	Pattern recognition reconstructs meaning	Traversal mutation and positional variation destabilize assumptions

3 Design Goals

3.1 Primary Goals

1. Human writable using pen and paper
2. Deterministic and reversible
3. Resistant to intuitive human parsing
4. Temporal evolution of mappings
5. Visually structured output
6. Cognitive ambiguity

3.2 Secondary Goals

1. Low computational requirement
2. Extensible keyspace
3. Standardized decoding behavior
4. Structured symbolic extensibility

4 Protocol Classification

STARLIGHT is an experimental symbolic encoding framework rather than a modern cryptographic system.

The protocol is intended for:

- ARG systems
- puzzle design
- fictional communication systems
- symbolic writing experiments
- interactive storytelling

STARLIGHT prioritizes:

- symbolic structure
- temporal transformation
- interpretive instability
- human-operable encoding

over computational security.

5 Notation and Symbols

For clarity, the following notation is used throughout this specification:

Notation	**Meaning**
B	Base Grid (key-dependent)
O_d	Operational Grid for weekday d (Monday–Sunday)
(r, c)	Coordinates in grid (row, column), 1-indexed
x_i	Plaintext symbol at position i
E_i	Encoded symbol after coordinate transfer
S_i	Positional shift value at position i
C_i	Final ciphertext symbol at position i
$\oplus$	Cyclic addition (forward shift)
$\ominus$	Cyclic subtraction (reverse shift)

6 Protocol Architecture

STARLIGHT consists of two layers:

6.1 Fixed Protocol Layer

The following components are universal and standardized:

- 7x7 symbol architecture (49 cells total)
- 7-day temporal cycle
- Deterministic reversibility
- Canonical ordering systems
- Fixed transformation philosophy

6.2 Variable Key Layer

The following components are configurable:

- Base Grid (manual construction)
- Temporal Seed (weekday/day of week)
- Initial Offset (optional position bias)

This separation ensures:

- protocol consistency
- interoperability
- extensibility
- identity preservation

7 Symbol Classes and Architecture

7.1 Grid Structure

The STARLIGHT encoding space consists of:

$$7 \times 7 = 49$$

cells allocated as follows:

Type	Count
Alphabet Symbols	33
Numbers	10
Structural Symbols	6
Total	49

7.2 Symbol Classes

7.2.1 Structural Symbols

The protocol includes the following structural symbols:

- **.** (period)
- **,** (comma)
- **?** (question mark)
- **!** (exclamation mark)
- **:** (colon)
- **;** (semicolon)

These symbols are treated as encodable structural operators rather than pure punctuation.

7.2.2 Extended Alphabet Symbols

STARLIGHT v1 extends the standard Latin alphabet using symbolic extension characters:

Symbol	Placeholder Name
SHF	Alpha
MIR	Beta
ORB	Gamma
NUL	Delta
ECH	Epsilon
SEA	Zeta
MASK	Omega

In STARLIGHT v1, extended symbols behave identically to standard alphabet symbols during:

- coordinate-transfer encoding
- positional shifting

- traversal ordering
- decoding

Extended symbols serve to:

- expand the symbolic encoding space
- increase visual diversity
- reserve symbolic capacity for future protocol extensions
- support future state-modifying semantics

Future protocol versions may assign operational semantics to these symbols.

8 Canonical Ordering and Cycles

All STARLIGHT implementations must preserve the following canonical ordering systems.

8.1 Alphabet Ordering

$$A \to B \to C \to \cdots \to Z$$

8.2 Numerical Ordering

$$0 \to 1 \to 2 \to \cdots \to 9$$

8.3 Structural Symbol Ordering

$$. \to , \to ? \to ! \to : \to ;$$

8.4 Extended Symbol Ordering

$$SEA \to ECH \to NUL \to ORB \to MIR \to SHF \to MASK$$

These orderings form the authoritative reference for all cyclic shifts, traversal operations, and canonical symbol sequencing throughout the protocol.

9 Canonical Shift Cycles

STARLIGHT v1 defines independent cyclic shift systems for each symbol class. Positive shifts move forward through the cycle. Negative shifts move backward through the cycle. All implementations must preserve identical cycle ordering to maintain interoperability.

Alphabet Cycle

$$A \to B \to C \to \cdots \to Z \to A$$

Number Cycle

$$0 \to 1 \to 2 \to \cdots \to 9 \to 0$$

Structural Symbol Cycle

$$. \to , \to ? \to ! \to : \to ; \to .$$

Extended Symbol Cycle

$$SEA \to ECH \to NUL \to ORB \to MIR \to SHF \to MASK \to SEA$$

9.1 Cyclic Shift Operations

9.1.1 Alphabet Characters

$$A - 1 = Z$$

$$Z + 1 = A$$

9.1.2 Numbers

$$0 - 1 = 9$$

$$9 + 1 = 0$$

10 Transport Delimiters

Transport delimiters are non-encodable structural markers used to organize ciphertext during transmission.

Unlike encodable symbols, transport delimiters do not participate in:

- coordinate-transfer encoding

- positional shifting
- traversal mutation
- temporal transformation

Instead, they preserve structural readability and transmission boundaries.

Common transport delimiters include:

Delimiter	Function
`<SPACE>`	Word boundary
`<ENTER>`	Line boundary
`<PAGE>`	Page separation
`<BLOCK>`	Traversal reset boundary
`<END>`	End-of-transmission marker
`::`	Traversal metadata boundary
`:`	Metadata token separator

Example transmission:

`TFN.8HC9H<SPACE>0,68U<SPACE>::<SPACE>7:M<ENTER>` where:

- `TFN.8HC9H 0,68U` is the traversal-ordered ciphertext
- `7:M` preserves original traversal ordering metadata

In this example:

- ciphertext symbols are encoded normally
- `<SPACE>` and `<ENTER>` remain unencoded
- transport delimiters preserve transmission structure

STARLIGHT distinguishes between:

- **Encodable symbols**, which participate in transformation and shifting
- **Transport delimiters**, which organize ciphertext structure but are not encoded

Transport delimiters exist outside the transformation pipeline and preserve logical structure during transmission.

11 Tokenization Rules

STARLIGHT v1 supports multi-character symbolic tokens.

These symbols are treated as indivisible encoding units during:

- coordinate-transfer encoding
- positional shifting
- traversal ordering
- metadata decoding

Current multi-character symbols include:

```
MASK, SHF, MIR, ORB, NUL, ECH, SEA
```

Tokenization follows greedy longest-match parsing.

Example:

```
MIRA
```

is parsed as:

```
MIR + A
```

rather than:

```
M + I + R + A
```

This guarantees deterministic symbol interpretation during encoding and decoding.

12 Base Grid System

12.1 Concept

The Base Grid is configurable and functions similarly to a cryptographic key.

The Base Grid itself is never directly used during encoding.

Instead:

$$\text{Operational Grid} = f(\text{Base Grid}, \text{Temporal State})$$

This ensures:

- no canonical alphabet exists
- day-one states are non-privileged
- mappings evolve continuously

12.2 Construction Purpose

The Base Grid functions as the foundational keyspace of STARLIGHT.

Unlike algorithmically generated grids, STARLIGHT grids are manually constructed by the user.

This preserves:

- personalization
- pen-and-paper usability
- memorability
- symbolic individuality

12.3 Construction Philosophy

STARLIGHT does not enforce a canonical arrangement.

Instead, the protocol defines:

- structural constraints
- validity rules
- stability requirements

while allowing users to design their own symbolic layouts.

This creates:

- unique keyspaces
- personalized encoding environments
- psychologically meaningful structures

12.4 Mandatory Construction Rules

12.4.1 Rule 1 — Unique Occupancy

Each of the 49 cells must contain exactly one unique element.

Duplicate entries are forbidden.

12.4.2 Rule 2 — Required Symbol Classes

Every valid Base Grid must contain:

Type	Count
Standard Alphabet Symbols	26
Numbers	10
Structural Symbols	6
Extended Alphabet Symbols	7
Total	49

12.4.3 Rule 3 — Extended Symbol Separation

Extended alphabet symbols should preferably avoid dense clustering.

Recommended spacing improves visual balance and reduces symbolic over-concentration within the grid.

Diagonal adjacency is permitted.

12.4.4 Rule 4 — Structural Symbol Stability

Structural symbols should preferably remain within interior regions of the grid.

This recommendation reduces traversal chaos and preserves structural readability.

12.4.5 Rule 5 — Nonlinear Arrangement Recommendation

Alphabetical ordering is strongly discouraged.

Example of discouraged arrangement:

A B C D E ...

Users are encouraged to create:

- asymmetric layouts
- symbolic clustering

- hidden geometric patterns
- emotionally meaningful placements
- intentionally irregular structures

This increases:

- personalization
- entropy
- resistance to intuitive reconstruction

12.5 Base Grid Design Considerations

When constructing a Base Grid, users should note:

- Nonlinear arrangement increases entropy and personalization
- Manual construction introduces human-directed randomness rather than algorithmic randomness
- Grid patterns should avoid obvious ordering (alphabetical, numerical sequence, etc.)
- Consider that grid structure may correlate with encoding quality

12.6 Grid Validity

A Base Grid is considered valid only if:

- all required symbol classes exist
- all entries are unique
- extended symbol spacing recommendations are respected
- the grid remains fully reversible under protocol transformations

Invalid grids may produce:

- unstable traversal
- decoding corruption
- irreversible transformations

12.7 Sample Base Grid

The following grid is a sample STARLIGHT Base Grid used throughout this specification.

It is intended only as a demonstration layout and does not represent a canonical configuration.

All operational grids used during encoding are derived from the Base Grid through temporal transformation.

M	7	A	?	T	SHF	2
X	;	L	0	R	!	ORB
C	9	MIR	E	B	:	Q
5	H	U	.	K	NUL	Y
G	,	P	ECH	3	D	V
W	F	1	J	8	S	MASK
Z	4	SEA	N	O	6	I

12.7.1 Design Notes

This sample grid demonstrates several STARLIGHT construction principles:

- No alphabetical ordering
- Extended Alphabet Symbols remain orthogonally separated
- Structural symbols remain mostly interior
- Symbol placement appears intentional but non-obvious
- Corners contain high-anchor symbols
- Human asymmetry is preserved

13 Temporal Architecture and Weekday Transformations

13.1 7-Day Temporal Cycle

STARLIGHT operates on a 7-day cyclic temporal basis aligned with the weekday calendar. This cycle determines how the Base Grid transforms into an Operational Grid for any given day of the week.

Each day applies a distinct, reversible geometric transformation to the Base Grid:

Day	Transformation
Monday	Horizontal Mirror
Tuesday	Vertical Mirror
Wednesday	Main Diagonal Reflection
Thursday	Anti-Diagonal Reflection
Friday	Clockwise Rotation
Saturday	Counterclockwise Rotation
Sunday	Center Reflection

This ensures:

- temporal state determinism
- human memorability
- full reversibility
- cyclic consistency

13.2 Transformation Examples

The following examples demonstrate how weekday transformations generate operational grids from the Base Grid. For simplicity, a reduced 3x3 demonstration grid is used. Actual STARLIGHT implementations operate on a 7x7 grid.

13.2.1 Base Demonstration Grid

A	B	C
D	E	F
G	H	I

13.2.2 Monday — Horizontal Mirror

Rows are mirrored horizontally.

Base Grid

A	B	C
D	E	F
G	H	I

Monday Operational Grid

C	B	A
F	E	D
I	H	G

13.2.3 Tuesday — Vertical Mirror

Rows are mirrored vertically.

Base Grid

A	B	C
D	E	F
G	H	I

Tuesday Operational Grid

G	H	I
D	E	F
A	B	C

13.2.4 Wednesday — Main Diagonal Reflection

The grid is reflected across the primary diagonal.

Base Grid

A	B	C
D	E	F
G	H	I

Wednesday Operational Grid

A	D	G
B	E	H
C	F	I

13.2.5 Thursday — Anti-Diagonal Reflection

The grid is reflected across the secondary diagonal.

Base Grid

A	B	C
D	E	F
G	H	I

Thursday Operational Grid

I	F	C
H	E	B
G	D	A

13.2.6 Friday — Clockwise Rotation

The entire grid rotates clockwise.

Base Grid

A	B	C
D	E	F
G	H	I

Friday Operational Grid

G	D	A
H	E	B
I	F	C

13.2.7 Saturday — Counterclockwise Rotation

The entire grid rotates counterclockwise.

Base Grid

A	B	C
D	E	F
G	H	I

Saturday Operational Grid

C	F	I
B	E	H
A	D	G

13.2.8 Sunday — Center Reflection

Sunday uses a center reflection transformation.

This preserves:

- cyclic stability
- visual symmetry
- reversibility
- operational simplicity

while avoiding direct exposure of the Base Grid.

Unlike an unmodified Base Grid state, the Sunday transformation prevents:

- plaintext-coordinate identity mapping
- direct structural leakage
- trivial symbol recovery

Base Grid

A	B	C
D	E	F
G	H	I

Sunday Operational Grid

I	H	G
F	E	D
C	B	A

This transformation creates a stabilized operational state while ensuring that symbols do not remain in their original positions.

Sunday therefore functions as:

- a calm reference state
- a symmetric reset state
- a non-trivial operational grid

within the STARLIGHT weekly cycle.

13.3 Transformation Philosophy

Weekday transformations are designed to be:

- visually intuitive
- reversible

- symmetric
- easy to perform manually
- cyclically stable

Transformations are always applied directly to the Base Grid. Operational grids never mutate recursively.

This guarantees:

- deterministic reversibility
- cyclic consistency
- protocol stability

14 Coordinate-Transfer Encoding

14.1 Concept

Encoding begins by locating the plaintext symbol inside the Base Grid. The symbol's coordinates are then transferred to the current Operational Grid.

Formally:

$$\text{BaseEncode}(x) = O_d(r_x, c_x)$$

where:

$$B(r_x, c_x) = x$$

and:

- B represents the Base Grid
- O_d represents the Operational Grid for weekday d

14.2 Example

Suppose the plaintext message contains the symbol:

$$H$$

and the current weekday state is Friday.

Inside the Base Grid (from Section 12.7):

$$H$$

is located at:

$$(4, 2)$$

Using the Friday Operational Grid (clockwise rotation):

$$(4, 2) = 0$$

Therefore:

$$H \to 0 \quad \text{(after coordinate transfer)}$$

15 Positional Shifting

15.1 Concept

After coordinate transfer encoding, the resulting symbol is shifted cyclically according to its position inside the message.

The shift pattern is:

$$0, +1, -1, +1, -1, +1, -1, \ldots$$

Meaning:

Position	Shift
1	0
2	+1
3	-1
4	+1
5	-1
...	...

This creates position-dependent encoding behavior.

15.2 Complete Encoding Formula

Let:

- x_i represent plaintext symbol at position i
- E_i represent coordinate-transfer encoded symbol
- S_i represent positional shift

Then:

$$E_i = O_d(r_i, c_i)$$

where:

$$B(r_i, c_i) = x_i$$

The final encoded symbol becomes:

$$C_i = \text{Shift}(E_i, S_i)$$

15.3 Example

Suppose coordinate transfer encoding produces:

$$K$$

and the symbol occupies position 2 in the message.

The second-position shift is:

$$+1$$

Therefore:

$$K + 1 = L$$

Final encoded symbol:

$$K \to L \quad \text{(after positional shift)}$$

16 Decoding Logic

Decoding reverses both stages.

1. Reverse positional shift

2. Locate symbol in Operational Grid
3. Transfer coordinates back to Base Grid

Formally:

$$E_i = \text{InverseShift}(C_i, S_i)$$

Then:

$$x_i = B(r_i, c_i)$$

where:

$$O_d(r_i, c_i) = E_i$$

17 Traversal Ordering

17.1 Purpose

The traversal system exists to disrupt automatic parsing behavior.

Visible order is not guaranteed to match interpretive order.

17.2 Traversal Rule

Words are processed using descending first-letter priority:

$$Z \to A$$

Higher-priority symbols appear earlier in the final ciphertext.

17.3 Traversal Priority System

Encoded words are arranged according to the first token of each encoded word using a deterministic priority system.

Priority classes are evaluated in the following order:

1. Alphabet symbols:
$$Z \to A$$
2. Numerical symbols:
$$9 \to 0$$

3. Extended alphabet symbols:

$$MASK > SHF > MIR > ORB > NUL > ECH > SEA$$

4. Structural symbols:

$$; > : > ? > ! > . > ,$$

If two encoded words share the same first-token priority, tie-breaking proceeds using:

1. descending lexicographic comparison
2. original traversal index as deterministic fallback

This guarantees stable traversal ordering while preserving deterministic reversibility.

17.4 Traversal Priority Values

Traversal ordering uses the following deterministic priority values. Higher numerical values are processed first.

Token Class	Priority Direction
Letters	Z → A
Numbers	9 → 0
Extended Symbols	MASK > SHF > MIR > ORB > NUL > ECH > SEA
Structural Symbols	; > : > ? > ! > . > ,

Equivalent numerical priority values may be assigned internally during software implementation.

Example implementation mapping:

Symbol	Priority Value
Z	1000
A	975
9	509
0	500
MASK	450
SHF	449
MIR	448
;	100
,	95

The exact numerical values are implementation-defined, provided ordering relationships remain stable.

17.5 Single-Letter Priority Rule

Single-letter words are always resolved first.

Examples:

I, A

This preserves sentence stability while maintaining traversal disruption.

17.6 Traversal Procedure

After encoding:

1. Inspect the first encoded character of each word
2. Compare the first characters using canonical priority rules
3. Rearrange words in descending order
4. Produce the final ciphertext sequence

17.7 Example

Visible sentence:

I SEE THE STARS TONIGHT

Traversal order:

1. I
2. TONIGHT
3. THE
4. STARS
5. SEE

18 Metadata System and Base-48 Indexing

18.1 Traversal Metadata Preservation

Traversal ordering may produce ambiguity when multiple encoded words are reordered during traversal processing.

To preserve deterministic reversibility, STARLIGHT v1 uses symbolic traversal metadata appended to the ciphertext after traversal ordering.

The metadata preserves:

- original encoded-word ordering
- traversal reconstruction
- deterministic decoding

Traversal metadata is encoded using the STARLIGHT Base Grid itself as a symbolic numbering system.

Instead of using Arabic numerals directly, STARLIGHT maps positional indices into symbolic representations derived from the Base Grid.

Metadata entries are separated using the transport delimiter:

:

The delimiter itself is not encodable and exists only at the transport layer.

18.2 Metadata Format

Example:

```
TFN.8HC9H 0,68U [7:M]
```

where:

- `TFN.8HC9H 0,68U` is traversal-ordered ciphertext
- `[7:M]` preserves original traversal ordering

Metadata tokens may contain multi-character symbolic units such as:

```
MIR, SHF, ORB, MASK
```

Therefore metadata decoding must use the same tokenization logic used during normal ciphertext parsing.

18.3 Metadata Reconstruction Procedure

Metadata reconstruction proceeds as follows:

1. split metadata by transport delimiter `:`
2. tokenize each metadata unit greedily
3. decode each symbolic index using Base-48 rules
4. reconstruct original traversal ordering

This preserves deterministic reversibility for arbitrarily long ciphertexts.

18.4 Base-48 Symbolic Indexing

STARLIGHT v1 represents traversal metadata using a symbolic positional numbering system derived from the Base Grid.

To avoid delimiter collisions during metadata parsing, the transport delimiter `:` is excluded from metadata digit generation.

Therefore the metadata numbering base becomes:

$$49 - 1 = 48$$

symbols.

All Base Grid symbols except `:` participate in metadata encoding.

This guarantees:

- parser-safe metadata decoding
- scalable symbolic indexing
- deterministic reconstruction
- delimiter isolation

The 48 symbols of the Base Grid (excluding `:`) act as the digit space of the numbering system.

If:

$$S_0, S_1, S_2, \ldots, S_{47}$$

represent the ordered symbols of the Base Grid, then positional metadata is encoded using:

$$n = \sum_{i=0}^{k} d_i \cdot 48^{k-i}$$

where:

- n is the original traversal index
- d_i represents the symbol value at position i

This permits:

- arbitrary traversal lengths
- scalable symbolic indexing
- protocol-native metadata representation

Examples:

Index	Symbolic Representation
0	M
1	7
2	A
47	I
48	7M
49	77
50	7A

The exact symbol ordering depends on the Base Grid configuration used by the implementation.

19 Parser Stability and Delimiter Isolation

Large ciphertext streams introduce parser ambiguity risks when transport delimiters overlap with encodable symbols.

STARLIGHT v1 avoids this through strict delimiter isolation.

Key stability rules:

- metadata exists only inside `[ ]`
- metadata separators are transport-only delimiters
- transport delimiters never participate in positional shifting
- metadata parsing occurs after ciphertext extraction
- tokenization uses greedy longest-match logic

These constraints prevent:

- metadata collisions
- traversal corruption
- decoding desynchronization
- delimiter ambiguity
- symbolic fragmentation

This ensures stable decoding behavior even for very large ciphertext streams.

20 Encoding Pipeline

STARLIGHT encoding proceeds through five deterministic stages:

Step	Operation
1	Tokenize plaintext using greedy longest-match parsing
2	Locate plaintext symbols inside the Base Grid
3	Transfer coordinates into the weekday Operational Grid
4	Apply positional cyclic shifts
5	Apply traversal ordering and append metadata

21 Decoding Pipeline

STARLIGHT decoding reverses all stages in reverse order:

Step	Operation
1	Extract metadata from inside `[ ]`
2	Decode symbolic traversal metadata
3	Restore original encoded-word ordering
4	Reverse positional cyclic shifts
5	Locate symbols in the Operational Grid
6	Transfer coordinates back to the Base Grid
7	Recover plaintext

22 Deterministic Reversibility

All STARLIGHT transformations must be:

- deterministic
- reversible
- state-reproducible

Without reversibility:

- decoding fails
- interoperability collapses
- protocol integrity is lost

Because all STARLIGHT transformations are deterministic and reversible, plaintext recovery is guaranteed when:

- the correct Base Grid is known
- the correct weekday state is known
- traversal metadata remains intact
- transport delimiters are preserved

23 Complete Encoding Walkthrough

This section demonstrates the complete STARLIGHT encoding process from plaintext through final ciphertext.

23.1 Setup

Plaintext Message:

```
HELLO STARLIGHT
```

Encoding Date: Friday

Base Grid: (using the sample grid from Section 12.7)

Friday Operational Grid (clockwise rotation of Base Grid):

C	X	M	5	G	W	Z
9	;	7	H	,	F	4
MIR	L	A	U	P	1	SEA
E	0	?	.	ECH	J	N
B	R	T	K	3	8	O
:	!	SHF	NUL	D	S	6
Q	ORB	2	Y	V	MASK	I

23.2 Step 1: Locate Coordinates in Base Grid

For each plaintext character, locate its position (row, column) in the Base Grid:

Character	Row	Column
H	4	2
E	3	4
L	2	3
L	2	3
O	7	5
S	6	6
T	1	5
A	1	3
R	2	5
L	2	3
I	7	7
G	5	1
H	4	2
T	1	5

Note: Word boundaries are preserved using transport delimiters such as `<SPACE>`. Transport delimiters are not encoded and exist outside the transformation pipeline (see Section 10).

23.3 Step 2: Transfer Coordinates to Operational Grid

Apply each coordinate to the Friday Operational Grid:

Plaintext Char	Base Grid Position	Operational Grid Value
H	(4, 2)	0
E	(3, 4)	.
L	(2, 3)	7
L	(2, 3)	7
O	(7, 5)	V
S	(6, 6)	S
T	(1, 5)	G
A	(1, 3)	M
R	(2, 5)	,
L	(2, 3)	7
I	(7, 7)	I
G	(5, 1)	B
H	(4, 2)	0
T	(1, 5)	G

Encoded values by word:

- HELLO → 0.77V

- STARLIGHT → SGM,7IB0G

23.4 Step 3: Apply Positional Shifting

Each character receives a position-dependent cyclic shift. The shift pattern across the entire message (including both words) is:

$$0, +1, -1, +1, -1, +1, -1, +1, -1, +1, -1, +1, -1, +1$$

Pos	Char	Class	Shift	Result
1	0	Number	0	0
2	.	Symbol	+1	, (next in cycle)
3	7	Number	-1	6
4	7	Number	+1	8
5	V	Letter	-1	U
6	S	Letter	+1	T
7	G	Letter	-1	F
8	M	Letter	+1	N
9	,	Symbol	-1	. (prev in cycle)
10	7	Number	+1	8
11	I	Letter	-1	H
12	B	Letter	+1	C
13	0	Number	-1	9
14	G	Letter	+1	H

Symbol cycle for structural symbols: . →, →? →! →:→; $(and reverse for negative shifts)$

Position 9: , minus 1 in cycle = .

Result after positional shifting:

- HELLO → 0,68U
- STARLIGHT → TFN.8HC9H

23.5 Step 4: Apply Traversal Ordering

Traversal reorders complete words by the first character of each encoded word, in descending alphabetical order:

Encoded Word	First Character
0,68U	0
TFN.8HC9H	T

Descending order: $T > 0$

Final Ciphertext (traversal-ordered words):

```
TFN.8HC9H 0,68U [7:M]
```

The word beginning with T appears first, followed by the word beginning with 0.

23.6 Final Verification

Stage	Result
Original Plaintext	HELLO STARLIGHT
Final Encoded Plaintext	TFN.8HC9H 0,68U [7:M]

24 Complete Decoding Walkthrough

This section demonstrates the complete reverse decoding process for the ciphertext generated in Section 23.

The purpose of this walkthrough is to demonstrate deterministic reversibility of the STARLIGHT protocol.

24.1 Ciphertext Input

The received ciphertext is:

```
TFN.8HC9H 0,68U [7:M]
```

where:

- `TFN.8HC9H 0,68U` is traversal-ordered ciphertext
- `[7:M]` is traversal metadata

The weekday state is:

```
Friday
```

and the same Base Grid from Section 12.7 is assumed.

24.2 Step 1 — Extract Metadata

The metadata exists only inside:

[ ]

Therefore:

`7:M`

is extracted.

The metadata delimiter is:

:

Splitting metadata produces:

`7  M`

These symbolic indices preserve original word ordering.

24.3 Step 2 — Decode Traversal Metadata

Using the Base-48 symbolic indexing system:

Metadata Symbol	Original Word Index
7	1
M	0

Therefore:

- `TFN.8HC9H` originally occupied position 1
- `0,68U` originally occupied position 0

24.4 Step 3 — Restore Original Traversal Order

Using the decoded indices:

$$0 < 1$$

the original encoded-word order becomes:

`0,68U TFN.8HC9H`

Traversal ordering has now been fully reversed.

24.5 Step 4 — Reverse Positional Shifting

The original shift pattern during encoding was:

$$0, +1, -1, +1, -1, +1, -1, +1, -1, +1, -1, +1, -1, +1$$

Decoding applies the inverse shift pattern:

$$0, -1, +1, -1, +1, -1, +1, -1, +1, -1, +1, -1, +1, -1$$

Applying inverse shifts:

Position	Ciphertext Symbol	Class	Inverse Shift	Result
1	0	Number	0	0
2	,	Structural	-1	.
3	6	Number	+1	7
4	8	Number	-1	7
5	U	Alphabet	+1	V
6	T	Alphabet	-1	S
7	F	Alphabet	+1	G
8	N	Alphabet	-1	M
9	.	Structural	+1	,
10	8	Number	-1	7
11	H	Alphabet	+1	I
12	C	Alphabet	-1	B
13	9	Number	+1	0
14	H	Alphabet	-1	G

Intermediate decoded words:

- `0.77V`
- `SGM,7IBOG`

These are the coordinate-transfer encoded forms before reversal.

24.6 Step 5 — Locate Symbols in Operational Grid

Using the Friday Operational Grid, each intermediate symbol is located.

Intermediate Symbol	Operational Grid Coordinate	Recovered Plaintext Symbol
0	(4,2)	H
.	(4,4)	E
7	(2,3)	L
7	(2,3)	L
V	(7,5)	O
S	(6,6)	S
G	(1,5)	T
M	(1,3)	A
,	(2,5)	R
7	(2,3)	L
I	(7,7)	I
B	(5,1)	G
0	(4,2)	H
G	(1,5)	T

24.7 Step 6 — Recover Plaintext

Reconstructing symbols in sequence produces:

```
HELLO STARLIGHT
```

The original plaintext has therefore been perfectly recovered.

24.8 Final Verification

Stage	Result
Original Plaintext	HELLO STARLIGHT
Final Decoded Plaintext	HELLO STARLIGHT
Match	YES

Decoding produces perfect plaintext recovery, confirming deterministic reversibility of the STARLIGHT v1 protocol.

25 Extended Encoding and Decoding Example

This section demonstrates a complete end-to-end STARLIGHT encoding and decoding cycle using the sample Base Grid and Friday temporal state with a longer message.

25.1 Plaintext Message

```
HELLO BRO HOW ARE YOU TODAY
```

Encoding weekday:

```
Friday
```

25.2 Encoding Steps Summary

25.2.1 Coordinate Lookup

Each plaintext symbol is located inside the Base Grid and its coordinates are noted.

25.2.2 Coordinate Transfer Encoding

Coordinates are transferred into the Friday Operational Grid (clockwise rotation).

25.2.3 Positional Shifting

STARLIGHT applies the shift sequence:

$$0, +1, -1, +1, -1, +1, -1, +1, -1, \ldots$$

Encoded words after shifting:

Plaintext Word	Encoded Word
HELLO	0V68U
BRO	Q.W
HOW	9W!
ARE	N.V
YOU	MW,
TODAY	HUTLO

25.2.4 Traversal Ordering

Words are reordered by first-token priority in descending order:

$$Q > N > M > H > 9 > 0$$

Therefore:

```
Q.W N.V MW, HUTLO 9W! 0V68U
```

25.2.5 Metadata Generation

Original word positions:

$$HELLO \to 0, \quad BRO \to 1, \quad HOW \to 2, \quad ARE \to 3, \quad YOU \to 4, \quad TODAY \to 5$$

After traversal reordering, metadata becomes:

$$[7 : A :? : T : M]$$

using Base-48 symbolic indexing.

25.3 Final Ciphertext

```
Q.W N.V MW, HUTLO 9W! 0V68U [7:A:?:T:M]
```

25.4 Decoding Verification

To decode:

1. Extract and decode metadata

2. Restore original traversal order
3. Reverse positional shifts
4. Locate symbols in the Operational Grid
5. Transfer coordinates back into the Base Grid
6. Recover plaintext

Recovered plaintext:

```
HELLO BRO HOW ARE YOU TODAY
```

26 Statistical Irregularity

STARLIGHT achieves resistance to frequency analysis through position-dependent cyclic shifting rather than explicit noise insertion. Each plaintext symbol produces different ciphertexts depending on its position in the message, preventing static monoalphabetic substitution patterns and eliminating predictable symbol repetition.

The shift-augmented encoding system introduces:

- position-dependent symbol variation
- reduced frequency consistency
- temporal variation
- reversible stateful encoding
- low mental overhead

Because identical plaintext symbols may produce different ciphertext symbols depending on position, STARLIGHT v1 avoids simple monoalphabetic substitution behavior.

27 Parser Disruption Behavior

The traversal system introduces:

- sentence-structure disruption
- positional ambiguity
- reduced grammatical predictability
- deterministic reversibility

- low computational overhead

Traversal ordering acts as a structural obfuscation layer within the STARLIGHT protocol.

28 Security Philosophy

STARLIGHT is not designed to compete with modern cryptographic standards.

It is designed to maximize:

- cognitive resistance
- interpretive instability
- state-dependent ambiguity
- symbolic depth

Its strength comes from:

- temporal evolution
- traversal disruption
- controlled ambiguity
- state-dependent interpretation

29 Limitations

STARLIGHT v1 operates under specific design constraints that users and implementers should understand:

29.1 Security Limitations

- **Not cryptographically secure:** STARLIGHT is not designed for high-security cryptographic applications and should not be used as a replacement for modern encryption standards.
- **Vulnerable to cryptanalysis:** Sophisticated frequency analysis, known-plaintext attacks, and grid-recovery techniques may compromise message security.
- **Limited keyspace:** The Base Grid construction rules and fixed temporal cycle limit the total keyspace compared to modern cryptographic systems.
- **Deterministic transformations:** Because weekday transformations are public and fixed, security depends entirely on Base Grid confidentiality.

- **Visible structural leakage:** Because transport delimiters remain unencoded, message structure such as word boundaries, line boundaries, and message length may remain partially observable.

29.2 Operational Limitations

- **Manual overhead:** Pen-and-paper encoding/decoding is time-consuming and error-prone compared to automated systems.
- **Grid memorization:** Users must memorize or protect the Base Grid, creating a single point of failure.
- **No error correction:** Single character errors in transmitted ciphertext cannot be detected or corrected automatically.
- **Limited message length:** Long messages become impractical for manual encoding.

29.3 Design Trade-offs

STARLIGHT v1 prioritizes:

- human usability over maximal entropy
- psychological resistance over mathematical security
- simplicity over complexity
- pen-and-paper feasibility over computational efficiency

This means STARLIGHT is optimally suited for puzzle systems, ARGs, and creative writing, but inappropriate for sensitive information protection.

30 Error Conditions

Decoding failure may occur if:

- traversal metadata is corrupted or incomplete
- incorrect weekday state is used during operational grid generation
- Base Grid differs between sender and receiver
- tokenization rules are violated during parsing
- transport delimiters are modified or removed

Typical failure symptoms include:

- irreversible traversal ordering (unable to restore original word sequence)
- symbol desynchronization (coordinate mismatch between grids)
- malformed token streams (invalid multi-character symbol boundaries)
- coordinate ambiguity (symbol not found in expected location)
- positional shift miscalculation (wrong cycle index applied)

When decoding fails, implementations should:

1. verify Base Grid integrity
2. confirm temporal state matches encoding date
3. validate metadata syntax and completeness
4. check for transport delimiter preservation
5. inspect for symbol tokenization errors

31 Complexity Characteristics

STARLIGHT v1 is designed for manual execution. Approximate operation complexity for message encoding/decoding:

Operation	Complexity
Coordinate Lookup	$O(n)$
Positional Shifting	$O(n)$
Traversal Sorting	$O(w \log w)$
Metadata Reconstruction	$O(w)$
Tokenization	$O(n \cdot m)$

where:

$$n = \text{total number of symbols in message}$$

and

$$w = \text{number of words}$$

and

m = maximum length of multi-character tokens

These complexities reflect manual pen-and-paper operations and assume single sequential processing passes.

32 Version Compatibility

STARLIGHT implementations must specify protocol version identifiers.

Example:

`STARLIGHT-v1`

Future versions may alter:

- traversal rules (ordering priorities, tie-breaking strategies)
- shift systems (cycle patterns, positional sequences)
- metadata encoding (symbolic representation, delimiter usage)
- temporal transformations (weekday rules, grid operations)
- extended symbol semantics (active protocol instructions)

Implementations using incompatible rulesets should be treated as distinct protocol families.

Current version is:

`STARLIGHT-v1`

33 Implementation Notes

Key implementation considerations include:

- preserve traversal metadata for deterministic decoding
- implement Base-48 symbolic traversal indexing (see Section 18.4)
- separate transport-layer metadata delimiters from encodable payload symbols
- maintain canonical ordering systems (see Section 8)

33.1 Practical Pen-and-Paper Considerations

STARLIGHT v1 is designed for pen-and-paper implementation. Users should:

- construct a 7×7 Base Grid using provided design rules (Section 12.7)
- memorize or keep the grid reference accessible
- perform coordinate lookups manually
- apply cyclic shifts using reference tables
- maintain traversal priority tables for consistent ordering

33.2 Software Implementation

- validate Base Grid construction rules before use
- precompute all 7 Operational Grids for a session
- implement cyclic shift functions for all symbol classes
- preserve traversal metadata for deterministic decoding
- separate transport delimiters from encodable payload symbols
- provide reversible decoding with identical machinery

34 Quick Reference Tables

34.1 Weekday Transformation Reference

Day	Transformation
Monday	Horizontal Mirror
Tuesday	Vertical Mirror
Wednesday	Main Diagonal Reflection
Thursday	Anti-Diagonal Reflection
Friday	Clockwise Rotation
Saturday	Counterclockwise Rotation
Sunday	Center Reflection

34.2 Positional Shift Pattern

The positional shift system follows:

$$0, +1, -1, +1, -1, +1, -1, \ldots$$

Position	Shift
1	0
2	+1
3	-1
4	+1
5	-1
6	+1
7	-1
...	...

34.3 Canonical Shift Cycles

Alphabet: $A \to B \to \cdots \to Z \to A$

Numbers: $0 \to 1 \to \cdots \to 9 \to 0$

Structural: $. \to , \to ? \to ! \to : \to ; \to .$

Extended: $SEA \to ECH \to NUL \to ORB \to MIR \to SHF \to MASK \to SEA$

34.4 Traversal Priority Order

Class	Order
Letters	Z > A
Numbers	9 > 0
Extended	MASK > SHF > MIR > ORB > NUL > ECH > SEA
Structural	; >: >? >! > . >,

34.5 Transport Delimiters

Transport delimiters remain unencoded.

Delimiter	Purpose
<SPACE>	Word boundary
<ENTER>	Line boundary
[ ]	Metadata container
:	Metadata separator

34.6 Multi-Character Tokens

MASK, SHF, MIR, ORB, NUL, ECH, SEA

Always parse using greedy longest-match.

35 Future Possibilities

STARLIGHT v1 establishes a foundational temporal encoding framework. Future extensions may include:

- adaptive traversal based on message statistics
- multi-grid chaining for longer communications
- recursive state memory linking previous symbols
- contextual shift patterns based on neighboring symbols
- spatial encoding layouts for visual writing systems
- state-modifying semantics for extended symbols

36 Conclusion

STARLIGHT is a temporal state-dependent encoding protocol designed to challenge automatic human parsing behavior.

Its defining features include:

- temporal state mutation through weekday-dependent grids
- traversal-based interpretation disruption
- position-dependent cyclic shifting
- deterministic reversibility
- structured symbolic ambiguity

- human-writable dynamic encoding

Rather than maximizing mathematical cryptographic security, STARLIGHT focuses on:

- psychological resistance to intuitive parsing
- interpretive instability
- symbolic depth and layering
- pen-and-paper usability

STARLIGHT v1 is optimally suited for:

- ARG systems and puzzle design
- fictional communication systems
- symbolic writing experiments
- interactive storytelling
- creative encoding challenges

The protocol emphasizes controlled chaos: all transformations are deterministic and reversible, yet produce ciphertext that resists pattern recognition and intuitive reconstruction. Through the combination of temporal state, coordinate transfer, positional variation, and traversal reordering, STARLIGHT creates a multi-layered encoding system that feels more like a symbolic framework than a simple cipher.

Future development of STARLIGHT should maintain the core principles of: - Deterministic reversibility - Temporal state evolution - Pen-and-paper usability - Psychological disruption of parsing assumptions - Human-directed randomness in grid construction

The specification presented here defines STARLIGHT v1 as a complete, functional protocol suitable for implementation, analysis, and extension. All technical details required for encoding, decoding, and verification are provided in full.

A Visual Reference Legend

This appendix provides a one-page compact reference for all core STARLIGHT v1 components.

A.1 Base-48 Metadata Indexing Reference

Using the sample Base Grid from Section 12.7:

Index	Symbol	Index	Symbol	Index	Symbol
0	M	16	MIR	32	D
1	7	17	E	33	V
2	A	18	B	34	W
3	?	19	Q	35	F
4	T	20	5	36	1
5	SHF	21	H	37	J
6	2	22	U	38	8
7	X	23	.	39	S
8	;	24	K	40	MASK
9	L	25	NUL	41	Z
10	0	26	Y	42	4
11	R	27	G	43	SEA
12	!	28	,	44	N
13	ORB	29	P	45	O
14	C	30	ECH	46	6
15	9	31	3	47	I

(The colon symbol is excluded from metadata indexing to prevent delimiter collision.)