Stream Cipher Blind Time Long Code (SC-BLTC) Protocol Specification

1. System Parameters

The following global parameters are predefined:

- F_s (Sampling Rate): 25 kHz. This is the time-domain resolution in simulation/implementation.
- R_c (Chip Rate): 5 kcps. This results in a signal bandwidth of approximately 5–6 kHz (depending on the RRC roll-off factor $\alpha = 0.25$). $T_c = 1/R_c$.
- OSF (Oversampling Factor): $OSF = F_s/R_c = 5$. OSF is fixed as an integer.
- SF (Spreading Factor): 1024. The processing gain per spread symbol is $G_p = 10\log_{10}(1024) \approx 30$ dB.
- k (Bits per Walsh Symbol): Fixed at $k = 8$.
- M_W (Walsh Orthogonal Set Size): $M_W = 2^k = 256$ (256-ary orthogonal modulation).
- R_{sym} (Spread Symbol Rate): $R_{\text{sym}} = R_c/SF = 5000/1024 \approx 4.88$ sym/s.
- R_{bit} (Post-spreading Bit Rate): $R_{\text{bit}} = k \cdot R_{\text{sym}} \approx 39.06$ bps (excluding preamble/pilot symbols and FEC overhead).
- R_{FEC} (Polar Code Rate): Used for Forward Error Correction, fixed at 1/2. The information rate is $R_{\text{info}} = R_{\text{bit}} \times R_{\text{FEC}}$ (excluding Header/CRC/padding overhead and preamble/pilot symbol overhead).
- K_{sec} (Shared Secret Key): A 256-bit key, pre-shared between the transmitter and receiver.
- T_{epoch} (Time Epoch): Defined as the Unix Epoch (1970-01-01 00:00:00 UTC).
- IV_{res} (Time Counter Resolution): Defines the time granularity for generating the spreading code seed, fixed at 1 ms.
- **Time Synchronization Assumption**: The Tx's local clock is roughly aligned with the Rx's time reference at the moment of transmission. Therefore, the Tx's TI_{tx} is guaranteed to fall within the Rx's blind search window $[t_{\text{rx}} - W,\ t_{\text{rx}} + W]$.
- M (FEC Codeword Length, coded bits): Fixed value $M = 512$.
- K (FEC Information Bit Length, uncoded bits): Fixed value $K = 256$, satisfying $K = M \cdot R_{\text{FEC}}$.
- **FEC**: Uses an Arikan Polar code with $(N, K) = (512, 256)$ and CRC-aided SCL (CA-SCL) decoding.
- **CRC**: A CRC-32C is appended inside U and is used both for post-decoding verification and for CA-SCL list selection.
- **Number of Preamble Symbols**: Fixed at $N_{\text{pre}} = 2$, used for acquisition and phase ambiguity resolution.
- **Number of Data Symbols**: $N_{\text{data}} = M/k = 64$.
- **Number of Pilot Symbols**: Fixed at $N_{\text{pilot}} = 16$. Pilots are used for continuous phase tracking under short coherence times of HF channels and to prevent frame-wide coherent demodulation failure due to cycle slips or phase flips.
- **Pilot Insertion Rule**: After the preamble, the N_{data} data symbols are divided into $N_{\text{blk}} = 16$ blocks of 4 symbols each. A pilot symbol is inserted at the beginning of each block. The structure of a single block is: `Pilot + Data + Data + Data + Data`.
- **Number of Spread Symbols per Frame**: $N_{\text{sym}} = N_{\text{pre}} + N_{\text{pilot}} + N_{\text{data}} = 82$.
- **Frame Length in Chips**: $L = N_{\text{sym}} \cdot SF = 82 \cdot 1024 = 83968$. The number of samples is $L \cdot OSF$.
- E_b/N_0 **Benchmark**: For a fixed $SF = 1024$, using $k = 8$ reduces the "number of chips allocated per bit", resulting in a $10\log_{10}(k) = 9$ dB processing gain loss. The receiver employs M_W-way orthogonal Walsh matched filtering, which provides a 3 dB equivalent advantage over binary decisions at the same E_b/N_0. The net change is a 6 dB degradation.
- **Walsh Code Definition**: Uses the row vectors $W_m[j] \in \{+1, -1\}$ of the order-$SF = 1024$ Sylvester-Hadamard matrix, where

$$W_m[j] = (-1)^{\text{popcount}(m \& j) \bmod 2}, \quad m \in [0, 1023],\ j \in [0, 1023]$$

This protocol only uses the first M_W orthogonal rows for $m \in [0, 255]$. $W_0[j] \equiv +1$.

2. Core Primitive: Cryptographically Secure Spreading Sequence Generator (CSPRNG Spreading)

AES-CTR mode is used as the Pseudo-Random Number Generator (PRNG).

Generation Function `GenCode(Key, TimeIndex, Length)`

Inputs:

- *Key*: The shared secret key.
- *TimeIndex*: An integer time counter for the transmission instant.
- *Length*: The number of chips to be generated.

Process:

1. **Construct Nonce/Counter**: Follows the standard CTR "Nonce + Counter" structure.
   - The 96-bit Nonce is defined as $Nonce = (TimeIndex \parallel Domain)$, where *TimeIndex* is 64-bit (endianness must be fixed in implementation), and *Domain* is a 32-bit constant `0x424C5443` (ASCII for "BLTC"), used for domain separation to prevent keystream reuse across different applications.
   - A 32-bit BlockCounter starts from 0 and increments for each block, forming the AES block input $IV_{\text{block}} = (Nonce \parallel BlockCounter)$.
2. **AES-CTR Generation**: Encrypt an all-zero data stream to produce the keystream bits.
3. **Mapping**: Map the output bitstream (0/1) to a bipolar chip sequence $(+1/-1)$.
   - **Endianness and Bit Order Convention**:
     - `TimeIndex` is encoded as **64-bit big-endian**.
     - `Domain` and `BlockCounter` are encoded as **32-bit big-endian**.
     - The AES-CTR output bytes are concatenated and then expanded into a bitstream with **bitorder=big** (MSB-first for each byte).
   - **Chip Mapping Rule**: `0 -> +1`, `1 -> -1`.
4. **Non-Reuse Constraint**: It is strictly forbidden to transmit twice with the same *TimeIndex* under the same shared key K_{sec}, as this would reuse the same CTR keystream. The transmitter powers down after each transmission and ensures the time interval between two consecutive transmissions is no less than IV_{res}, thus guaranteeing the uniqueness of *TimeIndex*.

Output: Spreading code sequence $C[n]$.

3. Transmitter Protocol

The transmitter is standalone; it does not query network time and relies solely on its local crystal oscillator.

> **Note**: The transmitter does not need network time synchronization, but the system assumes its local clock is roughly aligned with the receiver's time reference at the moment of transmission, as detailed in Section 1.

Step A0: Frame Format

To ensure the receiver can pre-determine the frame length and make decoding decisions, this protocol uses a fixed-length frame:

- **Information Bits (uncoded)**: U, with a length of $K = 256$ bits, composed of the following parts:
  1. **Header (plaintext/structural fields)**:
     - `Ver`: Protocol Version (4 bits).
     - `Type`: Message Type (4 bits).
     - `Len`: Payload Length (8 bits, in Bytes).
  2. **Payload**: Service data bits.
  3. **CRC**: A CRC-32C is computed over the Header+Payload for post-decoding verification.
  4. **Padding**: If Header+Payload+CRC does not fill up K bits, it is padded with zeros.

- **Polar Encoding**: U is encoded with a fixed code rate $R_{\text{FEC}} = 1/2$ into a codeword B of length $M = 512$ (coded bits) using a fixed Polar code $(N, K) = (512, 256)$.

  Frozen Set and Bit Mapping:

  1. Construct a length-N vector $\mathbf{u} = \{u_0, \ldots, u_{N-1}\}$ with a fixed frozen set $\mathcal{F}$ of size $N - K$. For all $i \in \mathcal{F}$, set $u_i = 0$. For all $i \notin \mathcal{F}$ (the information set), fill u_i with the next bit from U in increasing index order.
  2. The frozen set $\mathcal{F}$ is determined using Polarization Weight (PW) ranking for $N = 512$ with $\beta = 2^{1/4}$:

  $$w(i) = \sum_{j=0}^{\log_2 N - 1} b_j(i)\, \beta^j, \quad b_j(i) \in \{0, 1\} \text{ is bit } j \text{ of } i.$$

  The K indices with the largest $w(i)$ form the information set; all other indices are frozen.
  3. Apply the standard Arikan Polar transform to obtain the codeword $\mathbf{B} = \{b_0, \ldots, b_{N-1}\}$.

- **Full-Frame Interleaving**: Apply a fixed bit permutation $\pi(\cdot)$ over the full codeword to produce an interleaved sequence $B^{(\pi)} = \{b_0^{(\pi)}, \ldots, b_{M-1}^{(\pi)}\}$:

  $$b_j^{(\pi)} = b_{\pi(j)}, \quad \pi(j) = (A \cdot j + B) \bmod M, \quad A = 109,\ B = 37.$$

  This is a bijection because $M = 2^9$ and A is odd.

- **Symbol Mapping (256-ary Walsh)**: $B^{(\pi)}$ is grouped into $N_{\text{data}} = 64$ symbol indices $m_q \in [0, 255]$ of $k = 8$ bits each (MSB first):

$$m_q = \sum_{t=0}^{k-1} b_{qk+t}^{(\pi)} \cdot 2^{k-1-t}, \quad q = 0, \ldots, N_{\text{data}} - 1$$

These N_{data} indices are mapped to N_{data} orthogonal Walsh spread symbols.

- **Symbol Arrangement (Preamble + Pilots + Data)**:
  - Spread symbols are numbered $\ell = 0, \ldots, N_{\text{sym}} - 1$.
  - $\ell = 0$ and $\ell = 1$ are the two Preamble symbols.
  - Following the preamble are $N_{\text{blk}} = 16$ blocks, numbered $r = 0, \ldots, 15$. Each block consists of 5 spread symbols: 1 pilot + 4 data.
  - The position of the pilot symbol for block r is

  $$\ell_{\text{pilot}}(r) = 2 + 5r$$

  - The position of the q-th data symbol ($q = 0, \ldots, 63$) is determined as follows: let $r = \lfloor q/4 \rfloor$ and $s = q \bmod 4$, then

  $$\ell_{\text{data}}(q) = 2 + 5r + 1 + s$$

Step A: Payload Preparation

Assemble the frame according to Step A0 to get the fixed-length information bits U (including CRC and padding). Perform Polar encoding on U to get the codeword bit sequence $B = \{b_0, \ldots, b_{M-1}\}$. Apply the full-frame interleaver to get $B^{(\pi)}$, and pack it into a sequence of symbol indices $\{m_0, \ldots, m_{N_{\text{data}}-1}\}$ using $k = 8$ bits per symbol.

Step B: Determine Transmission Time Seed

Read the current local time t_{tx}. Calculate the time counter:

$$TI_{\text{tx}} = \left\lfloor \frac{t_{\text{tx}}}{IV_{\text{res}}} \right\rfloor$$

Note: This TI_{tx} is implicitly embedded in the signal, and the receiver must guess this value.

Step C: Generate Full-Frame Spreading Code

The total required chip length is $L = N_{\text{sym}} \times SF$.

Call the generation function:

$$C_{\text{seq}} = \texttt{GenCode}(K_{\text{sec}}, TI_{\text{tx}}, L)$$

Step D: Walsh Orthogonal Spreading Modulation

The cryptographic chip sequence is used as a "mask/scrambling code" and is combined with the orthogonal Walsh rows to generate the transmitted chips for each spread symbol.

The chips for the ℓ-th spread symbol ($\ell = 0, \ldots, N_{\text{sym}} - 1$) are denoted as $S[\ell \cdot SF + j]$, where $0 \le j < SF$.

- **Preamble Symbols** ($\ell = 0, 1$): Fixed to use a Barker-2 synchronization word with Walsh index 0, i.e., $[+W_0, -W_0]$:
  - **Preamble symbol 0** ($\ell = 0$, $+W_0$):

$$S[0 \cdot SF + j] = C_{\text{seq}}[0 \cdot SF + j]$$

  - **Preamble symbol 1** ($\ell = 1$, $-W_0$):

$$S[1 \cdot SF + j] = -C_{\text{seq}}[1 \cdot SF + j]$$

These two preamble symbols are used for acquisition and phase ambiguity resolution and do not carry information bits.

- **Pilot Symbols** ($\ell = \ell_{\text{pilot}}(r)$, $r = 0, \ldots, 15$): Fixed to use $W_0[j] \equiv +1$, therefore

$$S[\ell \cdot SF + j] = C_{\text{seq}}[\ell \cdot SF + j]$$

Pilot symbols are used for continuous phase tracking and cycle slip suppression and do not carry information bits.

- **Data Symbols** ($\ell = \ell_{\text{data}}(q)$, $q = 0, \ldots, N_{\text{data}} - 1$): Uses the m_q-th Walsh row. Let $\ell = \ell_{\text{data}}(q)$:

$$S[\ell \cdot SF + j] = W_{m_q}[j] \cdot C_{\text{seq}}[\ell \cdot SF + j]$$

This results in the final "chip sequence" S of length L at the chip-rate.

Step D2: Pulse Shaping and Oversampling

To constrain bandwidth and improve multipath resistance, the baseband chip sequence must be pulse-shaped.

- **Shaping Filter**: A Root-Raised-Cosine (RRC) filter $g_{\text{rrc}}(t)$ is used.
- **Oversampling**: The shaped waveform is output at a sampling rate of $F_s = OSF \cdot R_c$.

Let $S_{\text{wave}}[n]$ be the baseband waveform at the sampling rate, with a length of $L \cdot OSF$.

Step E: Transmission

Up-convert S_{wave} and transmit.

Step E2: Soft Shutdown

1. **Tail Padding**: After transmitting the N_{sym} symbols, append an additional $N_{\text{tail}} = 8$ all-zero symbols (Zero Padding) to allow the RRC filter's impulse response to decay naturally.
2. **Power Ramp-Down**: For the last L_{ramp} samples (recommended duration of about 20 ms), multiply the transmit waveform $S_{\text{wave}}[n]$ by a smooth-decaying window function $w[n]$ (from 1.0 down to 0.0) to prevent spectral splatter.
3. **Physical Shutdown**: After the digital waveform output has returned to zero, turn off the Power Amplifier (PA) in sequence, then turn off the Local Oscillator (LO) and system power after a 10 ms delay to avoid frequency drift artifacts.

4. Receiver Protocol

The receiver does not know TI_{tx}; it only knows that the transmission occurred sometime in the recent past. Assume the local time is t_{rx}, and the maximum clock drift and uncertainty window is $W = \pm 5$ seconds.

Step A: Signal Buffering

The receiver continuously records the signal into a circular buffer `Buffer`.

Step B: Signal Acquisition — Fast 2D Blind Acquisition via FFT

To achieve real-time blind demodulation, the receiver employs a "despread-then-FFT" frequency-domain search algorithm to avoid a time-domain brute-force search.

> The receiver first down-converts the signal to complex baseband I/Q. The acquisition stage operates directly on these **raw baseband samples** (no receive matched filter is required for acquisition). Matched filtering is applied later in the tracking/demodulation stage (Step C/D).

4.B.1 Search Space Definition

- **Time Hypothesis**: Iterate through all possible spreading code seeds $TI_{\text{search}} \in \left[\left\lfloor \frac{t_{\text{rx}}-W}{IV_{\text{res}}} \right\rfloor, \left\lfloor \frac{t_{\text{rx}}+W}{IV_{\text{res}}} \right\rfloor\right]$.
- **Intra-Epoch Starting Offset (Sample-Level)**: Since the transmitter quantizes time using $TI_{\text{tx}} = \lfloor t_{\text{tx}}/IV_{\text{res}} \rfloor$ and does not guarantee transmission starts exactly at the boundary of $TI_{\text{tx}} \cdot IV_{\text{res}}$, an unknown intra-epoch offset $\delta \in [0, IV_{\text{res}})$ exists for the same TI_{search}. To avoid missed detections, the receiver searches the offset at **sample-level resolution** over one IV:
  - Define $N_{\text{IV,samp}} = \text{round}(F_s \cdot IV_{\text{res}})$.
  - Search $n_{\text{offset,total}} \in [0, N_{\text{IV,samp}} - 1]$.
- **Frequency Offset Search**: The FFT is used to search CFO within a limited band $|\Delta f| \le f_{\text{search}}$, where $f_{\text{search}} = 8\,\text{kHz}$.

4.B.2 Fast Correlation Algorithm For each hypothesized TI_{search} and intra-epoch sample offset $n_{\text{offset,total}}$:

Let $n_{\text{base}}(TI_{\text{search}})$ be the start sample index (within the acquisition window) corresponding to the beginning of the IV interval for this TI_{search}.

0. Local Reference Construction (Two-Symbol Preamble)

Generate the corresponding spreading code C_{seq} from $(K_{\text{sec}}, TI_{\text{search}})$ with a length of $L = N_{\text{sym}} \cdot SF$ chips. Construct a two-symbol Barker-2 preamble chip sequence using the first $2SF$ chips:

$$S_{\text{pre}}[j] = C_{\text{seq}}[j], \quad 0 \le j < SF$$
$$S_{\text{pre}}[SF + j] = -C_{\text{seq}}[SF + j], \quad 0 \le j < SF$$

Apply the transmitter's RRC pulse shaping and sample at F_s to get the **TX-shaped** two-symbol preamble reference waveform $Ref_{\text{pre}}[k]$ of length $2SF \cdot OSF$.

1. Mixing/Despreading (Two-Symbol Coherent Integration Window)

Take a segment of the received raw baseband signal R and perform a point-wise conjugate multiplication with the local TX-shaped two-symbol preamble reference waveform Ref_{pre}. Let the total sample offset be $n_{\text{offset,total}}$, then

$$Z[k] = R[n_{\text{base}}(TI_{\text{search}}) + n_{\text{offset,total}} + k] \cdot Ref_{\text{pre}}[k]^*, \quad k = 0, \ldots, (2SF \cdot OSF) - 1.$$

Zero-pad Z to a fixed FFT length $N_{\text{FFT}} = 32768$.

2. Spectral Analysis

Perform an FFT on the mixed sequence Z:

$$P(f) = \left| \sum_k Z[k] e^{-j2\pi fk/F_s} \right|^2$$

The spectral peaks in the FFT output correspond to the locations of concentrated energy due to the residual frequency offset for that time hypothesis.

4.B.3 Peak Detection For each hypothesis $(TI_{\text{search}}, n_{\text{offset,total}})$ within the blind search window, define:

$$P_{\max}(TI_{\text{search}}, n_{\text{offset,total}}) = \max_{f \in \mathcal{B}} P(f), \quad \widehat{\Delta f}(TI_{\text{search}}, n_{\text{offset,total}}) = \arg\max_{f \in \mathcal{B}} P(f)$$

where $\mathcal{B} = [-f_{\text{search}}, +f_{\text{search}}]$ is the CFO search band.

Candidate Set Construction

The acquisition stage computes $P_{\max}$ and $\widehat{\Delta f}$ for **all** hypotheses $(TI_{\text{search}}, n_{\text{offset,total}})$ and retains the $K = 50$ hypotheses with the largest $P_{\max}$. Each retained hypothesis provides a candidate $(\widehat{TI}_{\text{tx}}, \hat{n}_{\text{coarse}}, \widehat{\Delta f}_{\text{coarse}})$.

Hybrid Integration Verification (Coherent Preamble + Noncoherent Pilots)

For each of the K candidates, perform the following steps and compute a final statistic Λ.

1. **CFO refinement (grid-loss elimination)**: refine the CFO estimate by a deterministic micro-search around the coarse estimate. For $f \in [\widehat{\Delta f}_{\text{coarse}} - 2\,\text{Hz},\ \widehat{\Delta f}_{\text{coarse}} + 2\,\text{Hz}]$ in steps of 0.25 Hz, compute the coherent two-symbol preamble metric

$$V_{\text{pre}}(f) = \left| \sum_{k=0}^{2SF \cdot OSF - 1} R[n_{\text{base}}(\widehat{TI}_{\text{tx}}) + \hat{n}_{\text{coarse}} + k] \cdot Ref_{\text{pre}}[k]^* \cdot e^{-j2\pi fk/F_s} \right|^2.$$

   Set $\widehat{\Delta f}_{\text{coarse}} = \arg\max_f V_{\text{pre}}(f)$ and $V_{\text{pre}} = V_{\text{pre}}(\widehat{\Delta f}_{\text{coarse}})$.

2. **Pilot noncoherent verification**: using $\widehat{\Delta f}_{\text{coarse}}$, compute a noncoherent pilot energy sum across all $N_{\text{pilot}} = 16$ pilots. For each pilot block r, construct a **pilot-specific** single-symbol TX-shaped reference waveform $Ref^{(r)}_{\text{pilot}}[k]$ as follows:

   - Let $\ell = \ell_{\text{pilot}}(r) = 2 + 5r$.
   - Take the corresponding SF chips from the local code: $S^{(r)}_{\text{pilot}}[j] = C_{\text{seq}}[\ell \cdot SF + j]$ for $j = 0, \ldots, SF - 1$ (since $W_0[j] \equiv +1$ for pilots).
   - Apply the transmitter's RRC pulse shaping and sample at F_s to obtain $Ref^{(r)}_{\text{pilot}}[k]$ of length $SF \cdot OSF$.

   For each pilot block $r = 0, \ldots, 15$, compute the nominal pilot position in samples:

$$\Delta_{\text{pos}}(r) = (2 + 5r) \cdot SF \cdot OSF.$$

   Then compute a timing-robust pilot correlation by searching a fixed local window of ± 32 samples around the nominal position:

$$C_r(\delta) = \sum_{k=0}^{SF\cdot OSF-1} R[n_{\text{base}}(\widehat{TI}_{\text{tx}}) + \hat{n}_{\text{coarse}} + \Delta_{\text{pos}}(r) + \delta + k] \cdot Ref_{\text{pilot}}[k]^* \cdot e^{-j2\pi\widehat{\Delta f}_{\text{coarse}}k/F_s}, \quad \delta \in [-32, 32].$$

where $Ref_{\text{pilot}}[k] \triangleq Ref^{(r)}_{\text{pilot}}[k]$ for the current pilot index r.

Accumulate the noncoherent pilot energy:

$$V_{\text{pilots}} = \sum_{r=0}^{15} \max_{\delta\in[-32,32]} |C_r(\delta)|^2.$$

3. **Final decision statistic**: compute

$$\Lambda = V_{\text{pre}} + V_{\text{pilots}}.$$

Compare Λ against a fixed threshold γ_{hybrid}. The value of γ_{hybrid} is determined offline by Monte Carlo simulation under complex AWGN at the receiver sampling rate using the full acquisition hypothesis volume and the complete hybrid verification procedure above, and is set such that the total false-alarm probability per 10-second scan window is $P_{\text{FA,window}} = 10^{-3}$.

If no candidate yields $\Lambda > \gamma_{\text{hybrid}}$, acquisition is declared a failure. Otherwise, select the candidate with the largest Λ and output $(\widehat{TI}_{\text{tx}}, \hat{n}_{\text{coarse}}, \widehat{\Delta f}_{\text{coarse}})$.

Time-Domain Correlation Refinement and RAKE Finger Initialization

Using the selected $(\widehat{TI}_{\text{tx}}, \hat{n}_{\text{coarse}}, \widehat{\Delta f}_{\text{coarse}})$, refine the timing offsets by coherent correlation with CFO compensation over a **multipath delay window** around $\hat{n}_{\text{coarse}}$.

Define an expected multipath delay span $W_{\text{multipath}}$ (e.g., 8 ms) and its half-width in samples:

$$N_{\text{search}} = \text{round}\left(F_s \cdot \frac{W_{\text{multipath}}}{2}\right).$$

Compute the refinement metric over a multipath delay window around $\hat{n}_{\text{coarse}}$, but clipped to valid nonnegative sample offsets. Define the absolute (nonnegative) sample-offset search range:

$$n_{\min} = \max(0, \hat{n}_{\text{coarse}} - N_{\text{search}}), \quad n_{\max} = \hat{n}_{\text{coarse}} + N_{\text{search}}.$$

In implementation, $n_{\max}$ is additionally clipped so that the correlation sum stays within the available sample window.

Compute:

$$Q(n) = \left| \sum_{k=0}^{2SF\cdot OSF-1} R[n_{\text{base}}(\widehat{TI}_{\text{tx}}) + n + k] \cdot Ref_{\text{pre}}[k]^* \cdot e^{-j2\pi\widehat{\Delta f}_{\text{coarse}}k/F_s} \right|^2, \quad n \in [n_{\min}, n_{\max}].$$

Let $\hat{n}_0$ be the maximizer of $Q(n)$. Set $N_{\text{finger}} = 3$ and initialize the RAKE receiver using the top-N_{finger} offsets according to $Q(n)$:

$$\{\hat{n}_{\text{finger},i}\}_{i=0}^{N_{\text{finger}}-1} = \text{Top-}N_{\text{finger}} \text{ indices of } Q(n).$$

The acquisition output parameters are: $\widehat{TI}_{\text{tx}}$, the main path starting position $\hat{n}_0$, the coarse frequency offset estimate $\widehat{\Delta f}_{\text{coarse}}$, and the set of RAKE finger starting positions $\{\hat{n}_{\text{finger},i}\}_{i=0}^{N_{\text{finger}}-1}$ (sorted ascending for implementation convenience).

4.B.4 Real-time Throughput Compute Estimate The following is an order-of-magnitude estimate for "real-time throughput" (processing delay is allowed, but backlog is not). System parameters: $F_s = 25\,\text{kHz}$, $R_c = 5\,\text{kcps}$, $SF = 1024$, $IV_{\text{res}} = 1\,\text{ms}$, blind search window $W = \pm 5\,\text{s}$.

- For each TI_{search}, an additional intra-epoch offset search is required at sample-level resolution over one IV: $N_{\text{IV,samp}} = \text{round}(F_s \cdot IV_{\text{res}}) = \text{round}(25\,\text{kHz} \cdot 1\,\text{ms}) = 25$ hypotheses.
- The number of TI_{search} values in a 10-second window is approximately $10\,\text{s}/1\,\text{ms} = 10\,000$. The total number of hypotheses is about $10\,000 \times 25 = 250\,000$.
- Each hypothesis requires processing $N = 2SF \cdot OSF = 2 \times 1024 \times 5 = 10240$ points. The FFT length is fixed at $N_{\text{FFT}} = 32768$ points. A single "point-wise despread + 32768-point FFT" operation is estimated at 3×10^6 FLOPs.
- Therefore, the total computation to scan a 10-second window is approximately $250\,000 \times 3 \times 10^6 \approx 7.5 \times 10^{11}$ FLOPs (about 750 GFLOPs). To achieve real-time throughput (processing a 10-second window in about 10 seconds), a sustained computational power of about $750\,\text{GFLOP}/10\,\text{s} \approx 75\,\text{GFLOP/s}$ is required.

Step C: RAKE Reception & Carrier Synchronization

After successful acquisition, the system switches to tracking and demodulation mode.

Using the acquisition outputs $(\widehat{TI}_{\text{tx}}, \hat{n}_0, \widehat{\Delta f}_{\text{coarse}}, \{\hat{n}_{\text{finger},i}\})$, first apply an initial CFO derotation to the raw baseband samples and then apply the receive RRC matched filter to form the matched-filtered sequence. All subsequent RAKE/DLL/PLL processing in Step C/D operates on this matched-filtered signal.

0. Full-Frame Local Code Generation

Use the acquired $\widehat{TI}_{\text{tx}}$ to call

$$C_{\text{seq}} = \texttt{GenCode}(K_{\text{sec}}, \widehat{TI}_{\text{tx}}, L)$$

to generate the full-frame cryptographic chip mask of length $L = N_{\text{sym}} \cdot SF$ for subsequent de-masking and Walsh demodulation.

1. RAKE Structure

The acquisition stage (4.B.3) outputs a set of RAKE finger starting positions $\{\hat{n}_{\text{finger},i}\}_{i=0}^{N_{\text{finger}}-1}$. The RAKE receiver assigns a "finger" to each of these N_{finger} components, aligning the initial code phase of each finger to the corresponding $\hat{n}_{\text{finger},i}$.

2. Dual-Loop Tracking

Due to the frame duration being on the order of seconds and the transmitter's unstable clock, both carrier frequency drift and chip clock drift must be overcome simultaneously.

- **a. Carrier Tracking (PLL/Costas Loop)**: Initialize with the coarse frequency offset $\widehat{\Delta f}_{\text{coarse}}$ from the acquisition stage, and track the residual intra-frame frequency offset and phase jitter (represented by the symbol-rate NCO phase θ_ℓ).
  - **Implementation Requirement**: A small closed-loop PLL/Costas loop must be run at the symbol rate. The loop state must be continuously updated even in the intervals between any two adjacent pilot blocks (i.e., within the 4 data symbols of each pilot block). Open-loop approximations like "pilot phase linear fitting / block-level phase compensation" must not be used as a substitute.

  Define the symbol-rate NCO state with phase θ_ℓ (rad) and phase increment ω_ℓ (rad/symbol), updated using a second-order Type-II loop:

$$e_\ell = \text{atan2}(\Im\{z_\ell\}, \Re\{z_\ell\}), \quad \omega_{\ell+1} = \omega_\ell + K_i e_\ell, \quad \theta_{\ell+1} = \theta_\ell + \omega_{\ell+1} + K_p e_\ell.$$

  The input to the error detector, z_ℓ, is chosen as:

  - **Preamble/Pilot (known W_0)**: $z_\ell = \sum_{j=0}^{SF-1} \mathbf{u}_\ell[j]$;
  - **Data Symbol (decision-directed)**: First, make a Walsh decision $\hat{m}_\ell = \arg\max_{m \in [0,255]} |R_\ell[m]|$, then use $z_\ell = R_\ell[\hat{m}_\ell]$.

Symbol-Rate Frequency-Hypothesis Bank

Because T_{sym} can be large (e.g., $\approx 205\,\text{ms}$), even sub-Hz residual Doppler causes significant phase rotation within a single symbol. In low SNR, a pure phase-detector PLL can therefore suffer cycle slips. Instead of a within-symbol FLL, use a small brute-force frequency bank around the current PLL estimate at the symbol rate.

Let $f_{\text{PLL},\ell} = \omega_\ell/(2\pi\, T_{\text{sym}})$ (Hz). For each symbol ℓ:

1. **Hypotheses**: generate $f_k = f_{\text{PLL},\ell} + \Delta f_k$ with $\Delta f_k \in \{-4, -3.75, \ldots, +4\}\,\text{Hz}$ (step $0.25\,\text{Hz}$), and map to $\omega_k = 2\pi f_k\, T_{\text{sym}}$.
2. **Per-hypothesis demod**:
   - **Preamble/Pilot (W_0 known)**: select $k_{\text{best}} = \arg\max_k \left|\sum_{j=0}^{SF-1} \mathbf{u}_{\ell,k}[j]\right|$.
   - **Data**: for each k, compute $R_{\ell,k}[m]$ via a 1024-point FHT and select $(k_{\text{best}}, \hat{m}_\ell) = \arg\max_{k,m\in[0,255]} |R_{\ell,k}[m]|$.
3. **PLL update**: always advance θ by the selected $\omega_{\ell,k_{\text{best}}}$, then apply the phase-detector correction using z_ℓ (pilot/preamble prompt sum or $R_{\ell,k_{\text{best}}}[\hat{m}_\ell]$).
4. **Frequency snap (anti-slip)**: if the same non-zero $\Delta f_{k_{\text{best}}}$ bin is selected for N consecutive symbols (e.g., $N = 3$) with sufficient confidence, treat the PLL center as mis-tracked and correct ω_ℓ by snapping it toward the bank peak (equivalently, shift f_{PLL} by $\Delta f_{k_{\text{best}}}$).

Loop Gain Design (Discrete-Time)

The loop gains (K_p, K_i) should be designed in discrete time (as a function of the update period T_{sym}). Small-T analog approximations like $K_p \approx 2\zeta\omega_n T$ and $K_i \approx (\omega_n T)^2$ should not be used when $\omega_n T$ is not small.

A standard choice is to specify loop bandwidth B_L (Hz) and damping factor ζ, set $\omega_n = 2\pi B_L$, and map the continuous-time poles to discrete time via $z = \exp(sT)$:

$$r = \exp(-\zeta\omega_n T), \quad \phi = \omega_n\sqrt{1-\zeta^2}\,T, \quad z_{1,2} = re^{\pm j\phi}.$$

For the update equations above, the corresponding stable discrete-time gains are:

$$K_p = 1 - r^2, \quad K_i = 1 + r^2 - 2r\cos\phi.$$

- **b. Code Timing Tracking (DLL - Delay Locked Loop)**: Use an Early-Late Gate to track the expansion/contraction of the chip clock (Code Doppler). Due to the transmitter's crystal oscillator drift, a cumulative drift of several chips can occur over a transmission of more than ten seconds. Failure to track this will lead to a mismatch of the spreading sequence. The DLL dynamically adjusts the sampling instant to keep the correlation peak in the "Prompt" channel.

3. Despreading and Combining

For each spread symbol ℓ and each finger i, perform the following operations and combine in the chip domain:

1. Starting from $\hat{n}_{\text{finger},i}$, extract the received segment $Y_{i,\ell}$ for this symbol.
2. Compensate for the carrier phase estimated by the PLL/Costas loop: $Y'_{i,\ell}(t) = Y_{i,\ell}(t)\cdot e^{-j\theta_\ell}$.
3. After matched filtering and DLL alignment, extract SF chip samples at the chip instants to get the chip vector $\mathbf{y}_{i,\ell}[j]$.
4. De-mask using the locally generated cryptographic chip mask: $\mathbf{u}_{i,\ell}[j] = \mathbf{y}_{i,\ell}[j]\cdot C_{\text{seq}}[\ell\cdot SF + j]$.

All fingers are combined using Maximal-Ratio Combining (MRC) by a weighted sum in the chip domain to obtain the de-masked chip vector for that symbol:

$$\mathbf{u}_\ell[j] = \sum_i w_i \cdot \mathbf{u}_{i,\ell}[j], \quad w_i = \frac{A_i^*}{\sum_k |A_k|^2}$$

where A_i is the estimated complex channel gain for the i-th finger. A_i is initialized by coherent averaging across the two preamble symbols ($\ell = 0, 1$) and must be dynamically updated using the intra-frame pilots; the preamble weights must not be used for the entire frame. It is recommended to update the weights with a time constant of "every 16 pilot blocks" (equivalent to smoothing over the last $N_{\text{pilot}} = 16$ pilots):

$$\tilde{A}_i^{(r)} = \frac{1}{SF} \sum_{j=0}^{SF-1} \mathbf{u}_{i,\ell_{\text{pilot}}(r)}[j] \cdot e^{-j\theta_{\ell_{\text{pilot}}(r)}}, \quad A_i^{(r)} = (1-\alpha)A_i^{(r-1)} + \alpha\tilde{A}_i^{(r)}, \quad \alpha = \frac{1}{16}.$$

Implementation Note:

- **Fractional Sampling**: When the DLL/code phase requires sub-sample level adjustment, linear interpolation can be performed on the sample sequence after the matched filter to obtain the equivalent sample value at the chip instant.

Step D: Walsh Demodulation and Polar (CA-SCL) Decoding

1. Pilot-aided Carrier Tracking

The coherence time of HF channels is often significantly shorter than the frame duration. Therefore, this protocol periodically inserts pilot symbols within the frame, which are used to aid the closed-loop PLL/Costas in Step C to continuously track the carrier phase/residual frequency offset. In the intervals of 4 data symbols between two pilot blocks, the loop continues to update in a decision-directed manner to avoid intra-block loss of coherence.

2. Walsh Matched Filtering (FHT)

For each data symbol $\ell = \ell_{\text{data}}(q)$ ($q = 0, \ldots, N_{\text{data}} - 1$) and each symbol-rate frequency hypothesis k (selected from the bank in Step C), compute the M_W-way correlation on the carrier-compensated chip vector:

$$R_{\ell,k}[m] = \sum_{j=0}^{SF-1} \mathbf{u}_{\ell,k}[j] \cdot W_m[j], \quad m \in [0, 255]$$

Use a 1024-point Fast Walsh-Hadamard Transform (FHT) to compute the full correlation for $m \in [0, 1023]$, and then truncate to the first M_W results for $m \in [0, 255]$. Select the hypothesis k_{best} for this symbol using the peak energy criterion (Step C), and pass $R_{\ell,k_{\text{best}}}[m]$ to the soft-demapper.

3. Soft Information (LLR) Generation

Apply a decision-directed common phase rotation before soft-demapping to make the decided peak real-positive:

$$\widetilde{R}_\ell[m] = R_{\ell,k_{\text{best}}}[m] \cdot e^{-j \arg\left(R_{\ell,k_{\text{best}}}[\hat{m}_\ell]\right)}.$$

Let the metric be $D_\ell[m] = \Re\{\widetilde{R}_\ell[m]\}$ (or $D_\ell[m] = \Re\{R_{\ell,k_{\text{best}}}[m]\}$ if no additional rotation is used). For the t-th bit within each symbol ($t = 0, \ldots, k-1$), calculate the max-log LLR:

$$LLR_{qk+t} = \max_{m:\,((m \gg (k-1-t))\&1)=0} D_\ell[m] - \max_{m:\,((m \gg (k-1-t))\&1)=1} D_\ell[m]$$

The $N_{\text{data}} \cdot k = M$ LLRs correspond to the interleaved coded bits $B^{(\pi)}$. Apply the inverse permutation to deinterleave them before decoding:

$$LLR_i = LLR^{(\pi)}_{\pi^{-1}(i)}.$$

4. CA-SCL Polar Decoding and Output

Perform CRC-aided SCL (CA-SCL) Polar decoding on the deinterleaved LLRs to obtain the information bits $\hat{U}$. Use the existing CRC-32C (embedded inside U) as the list selection rule: choose the best-metric candidate that passes CRC; if none pass CRC, choose the best-metric candidate. Then perform a final CRC-32C check; if it passes, output $\hat{U}$ and truncate the payload according to the `Len` field in the Header. If it fails, discard the packet.