Full Spectrum Radio: A Jam-Resistant, Digital Entropy-Driven Communication System

Enrique Flores

July 24, 2025

Abstract

This document outlines a novel wireless communication protocol known as **Full Spectrum Radio (FSR)**. Unlike traditional frequency-hopping or spread-spectrum methods that rely on pseudorandom number generators (PRNGs), FSR utilizes digital entropy generated via a high-speed digital random bit generator (MKRAND). The resulting unpredictability renders the signal effectively untraceable and unjammable using conventional means. Synchronization and recovery mechanisms are proposed to ensure resilience across adverse environments.

In addition to its entropy-based stealth characteristics, FSR enables **encrypted communication between paired peers**, allowing for secure transmission of sensitive messages and protected key exchange. This layer of peer-authenticated encryption ensures that even within the unpredictable frequency space, trust and confidentiality can be preserved across distributed mesh participants.

1 Introduction

Wireless communication has historically evolved to cope with noise, interference, and increasingly, hostile environments where jamming and signal interception are real threats. One of the most effective countermeasures to these threats has been the development of **frequency hopping spread spectrum** (FHSS) and other forms of spread spectrum communication. In such systems, a transmitter rapidly switches between frequencies within a designated band according to a known sequence, making it more difficult for adversaries to jam or intercept the signal.

Traditional spread spectrum systems rely on **pseudorandom number generators (PRNGs)** to determine the hopping sequence. While PRNGs can produce sequences that appear random, they are ultimately deterministic. Given partial knowledge of the internal state, an attacker can infer past and future frequency hops. This vulnerability undermines the robustness of the communication channel in adversarial scenarios, especially in military, intelligence, and police applications.

The limitations of PRNG-based systems create a strong need for communications protocols that offer true **unpredictability**. In such a system, frequency selection must be driven

not by algorithms with discoverable states, but by a source of true entropy—randomness that cannot be anticipated, reproduced, or reverse-engineered by unauthorized devices, even ones implementing this protocol. This shift from pseudorandom to **digital entropy-derived** frequency selection represents a fundamental security upgrade.

Full Spectrum Radio (FSR) introduces a new paradigm in secure wireless communication: by deriving frequency hops from a **hardware-based digital random bit generator** (MKRAND), each transmission burst becomes uniquely untraceable and inherently immune to predictive jamming. Furthermore, by spreading signal energy across the entire available spectrum in a uniformly unpredictable manner, FSR makes detection via spectrum analysis significantly harder, reducing the electronic signature of the transmission itself.

This document outlines the architecture, principles, and potential applications of Full Spectrum Radio, including synchronization methods, signal structure, and implementation pathways using modern programmable hardware. FSR is designed for the future of resilient, stateless, stealthy communication.

2 Threat Model

FSR is designed for operation in hostile RF environments with the following threat assumptions:

- **Wideband Spectrum Surveillance:** Adversaries are capable of scanning large swaths of RF space in real time using digital receivers and ML-based classifiers.
- **Reactive Jamming:** Adversaries can respond to detected transmissions within milliseconds, attempting to disrupt active links by flooding the same frequency bin.
- **Protocol Fingerprinting:** Adversaries can infer metadata or structure from transmission patterns or signal headers.
- **Key Material Capture:** Adversaries may recover transmitted data bursts, but lack access to the genesis block or pairing keys.

3 System Overview

3.1 Components

The Full Spectrum Radio (FSR) system is built from a small number of tightly integrated components, each of which contributes to the overall unpredictability, stealth, and resilience of the communication channel.

- **Entropy Source (MKRAND)** — At the heart of FSR is MKRAND, a deterministic high-speed digital entropy generator. Unlike traditional random number generators that depend on environmental noise, clock drift, or quantum effects, MKRAND derives its entropy entirely from a shared digital seed known as the *genesis block*. Devices initialized with the same genesis block produce identical entropy streams, which are

used to select matching frequency bins. This allows perfect synchronization without requiring external clocks, timestamps, or environmental cues.

- **Frequency Selection Mechanism** — The frequency hopping logic consumes 128-bit blocks from MKRAND and uses a defined subset of those bits to select a transmission frequency within the system's configured RF band. For example, if the usable band is divided into 256 subchannels, only 8 bits of the entropy block are required to select a slot.

  This design ensures that:

  - Even if an adversary captures one or more transmitted bursts, they gain no information about the next hop location.
  - The remaining bits of the 128-bit block may be used for temporal spreading (e.g., burst timing within a window), modulation parameters, or embedded sync hints—providing multidimensional entropy without any reuse.
  - Knowledge of the circuit design or frequency binning algorithm is insufficient to predict future hops without possession of the genesis block.

- **Transmitter and Receiver Architecture** — Both ends of the communication channel are equipped with identical logic for entropy consumption and frequency selection. The transmitter draws digital entropy from MKRAND to schedule its next burst, while the receiver—initialized with the same genesis block—uses the same entropy stream to anticipate hop positions. This approach eliminates the need for time synchronization or explicit handshaking. Instead, the system relies solely on a shared **genesis block**, which acts as a deterministic seed and serves the role of a cryptographic key. Without knowledge of this block, an adversary cannot reconstruct or predict the sequence.

3.2 Design Goals and Differentiators

The Full Spectrum Radio protocol is designed to operate under contested, resource-constrained, and decentralized conditions. It prioritizes the following properties:

- **Resistance to Detection and Jamming** — The use of digital entropy-derived frequency hopping ensures that transmission patterns are indistinguishable from background noise and vary unpredictably between bursts. This makes it extremely difficult for adversaries to detect or jam the signal, even with reactive spectrum analysis.

- **No Predictable Pattern** — Unlike pseudorandom number generators (PRNGs), which can leak internal state or repeat under scrutiny, MKRAND (a digitally seeded random bit generator) produces a stream of entropy that cannot be reverse-engineered. Frequency selections use only a subset of each entropy block, so observing one burst reveals nothing about past or future activity.

- **Stateless or Re-synchronizable Communication** — The protocol tolerates packet loss, burst failure, and transient disconnects without requiring session tracking. As long as both parties share the same genesis block, receivers can resynchronize with transmitters by realigning to the next successful burst.

- **Minimal Overhead** — Frequency, timing, and addressing are all derived directly from the entropy stream. No explicit headers, key exchanges, or synchronization metadata are needed. This lean encoding maximizes spectrum efficiency and reduces interceptability.
- **Scalable Swarm Coordination** — Devices subscribe to short, tag-based identifiers instead of maintaining pairwise connections. This enables thousands of units to operate in the same band, listening for relevant data while ignoring irrelevant traffic — ideal for drone swarms or sensor mesh applications.
- **Hardware Simplicity and Portability** — All protocol logic is designed to fit within a small FPGA or microcontroller. No operating system, floating-point math, or external clock synchronization is required, enabling deployment in edge devices and austere environments.

4 Digital Entropy-Based Frequency Selection

FSR uses digital entropy derived from MKRAND to deterministically select frequency hopping bins.

- **Entropy Block**: A 128-bit output from MKRAND used to derive frequency hops.
- **Block Height**: A 64-bit counter indicating the number of blocks since the genesis block.
- **Current Block**: The chain state, or present entropy block at a given block height.

4.1 Hop Derivation from Entropy Blocks

Each 128-bit entropy block is subdivided into multiple non-overlapping hop segments. For example, with $N = 10$ bits per frequency bin, each block yields $\left\lfloor \frac{128}{10} \right\rfloor = 12$ discrete hops.

This approach significantly reduces the required entropy throughput, enabling high hop rates (e.g., 100 kHz hopping from an 8.3 kHz block generation rate).

The number of bits per hop segment—and the interpretation of each segment as a bin index—are implementation-defined. The mapping from entropy to RF frequency is application-specific and may vary based on regulatory constraints (e.g., FCC, CE), operational band, and hardware capabilities (e.g., SDR or FPGA front ends). This abstraction ensures that FSR remains adaptable across diverse environments, from ISM-band mesh radios to mmWave military networks.

4.2 Encrypted Sync Block Forwarding

Rather than sharing the current entropy block explicitly — which would compromise the same cryptographic guarantees as the genesis block itself — FSR enables synchronization via **secure block forwarding**. Upon detecting a join beacon within the rendezvous region, any

mesh participant configured as a **sentinel node** encrypts a *future* entropy block, or chain state, such as B_{N+5}, using a pre-shared pairing key provisioned during group onboarding.

This encrypted sync block is then transmitted over a known rendezvous frequency. The joining device decrypts the message, seeds its MKRAND generator with the received entropy block, computes the corresponding bin, waits for the message burst, and begins frequency hopping in perfect alignment with the mesh. The accompanying **block height** (transmitted in plaintext) allows the new participant to align its local blockchain clock and participate in time-based messaging.

- The joiner requires only a single valid decrypted sync message from a sentinel — no persistent session or handshake is necessary.
- Any valid message from any node can serve as a sync vector; no special SYNC frame type is required.

To minimize the number of pairing keys required, a group of authorized devices may share a common **sentinel block** — a 128-bit symmetric key provisioned during deployment. Any node with this key can securely onboard new participants, and all joiners provisioned with the same key will recognize and decrypt sync messages without additional negotiation.

4.3 Future-Time Computation and Long-Term Synchronization

Once a node is actively participating in the mesh and has synchronized to the shared entropy stream, it may wish to reference or schedule events far into the future — beyond a small number of immediate blocks. To do this, FSR employs a deterministic, integer-based method of future-time calculation that avoids floating point precision errors while supporting arbitrarily large time horizons.

The process is as follows:

1. The node determines the desired delay in real-world time units (e.g., milliseconds, seconds, hours, or even centuries).
2. It divides this interval by the known **block period** (i.e., time between entropy blocks) to compute the required number of blocks, rounding down to the nearest whole block.
3. This block delta is then added to the current **block height** to yield a *future block height* B_f.
4. Optionally, an intra-block offset (expressed in milliseconds or ticks) can be appended to improve temporal resolution.

This mechanism enables any node to encode future timestamps that can be interpreted with crystal-clock precision across the entire mesh — regardless of geographic dispersion or uptime duration. For example, a swarm could schedule a coordinated maneuver 500 years in the future (given stable entropy generation) with millisecond alignment, all using simple integer arithmetic. The only assumptions are a shared understanding of block timing and

synchronized entropy source initialization, both of which are ensured via the onboarding protocol and ongoing mesh maintenance.

This allows FSR-enabled swarms to execute synchronized behaviors, delayed activations, or deferred message delivery with deterministic accuracy — enabling use cases ranging from deep-space coordination to cyber-resilient battlefield latency.

4.4 Coexistence and Collision Handling

Figure 1: Frequency-hopping bursts for two transmitters over 128 time slots. Blue bars represent bursts from transmitter A, green bars represent bursts from transmitter B, and red bars indicate collision events where both transmitters occupy the same slot.

FSR operates in a decentralized, stateless manner, where each transmission hop is determined independently by a shared entropy stream derived from the genesis block. As such, occasional **frequency collisions**—where two or more independent FSR networks (with distinct genesis blocks) select the same frequency bin at the same time—are not only possible but expected in large-scale deployments.

No special mechanism is required to detect or resolve such collisions. Because transmission bursts are short and the entropy stream immediately advances to a new frequency bin, all participants **naturally diverge** after a single collision. This property makes FSR highly resilient to transient interference and eliminates the need for retransmission protocols at the physical layer.

However, higher-layer protocols may optionally implement additional logic to improve coordination or recovery in high-collision environments:

- **Listen-Before-Talk (LBT):** Nodes may perform a quick energy scan prior to transmission to avoid busy bins.
- **Side-channel Coordination:** A dedicated bin or low-rate beacon may be used to communicate mesh identity or presence in shared bands.

By default, FSR treats frequency collisions as benign events, relying on the forward-only hop schedule and loss-tolerant communication to ensure long-term stability and throughput.

5 Synchronization and Secure Onboarding

5.1 Block Height and Timing

The Full Spectrum Radio protocol is built atop a deterministic random bit generator (MKRAND), where each entropy block is generated every fixed number of clock cycles—typically 256. This makes the **block height** a natural representation of elapsed time within the mesh, with each increment corresponding to a fixed, known duration (e.g., 256 cycles at 100 MHz = 2.56 μs).

To simplify coordination and enable relative time calculations, FSR defines a 64-bit **block height counter**:

- It represents the current numeric index of the entropy stream (i.e., the number of blocks since the genesis block).
- It is safe to transmit in plaintext, as it reveals nothing about the genesis block or chain state.
- It enables devices to express timeouts, message timestamps, and synchronization windows without relying on wall-clock time.
- Its 64-bit range provides an uptime of over 9.2 quintillion blocks, or approximately 7500 years at 2.56 μs per block:

$$\frac{2^{64} \times 2.56\,\mu s}{60 \times 60 \times 24 \times 365} \approx 7455 \text{ years}$$

- Rollover is safe and does not affect operation, provided relative comparisons are performed modulo 2^{64}.

5.2 Tradeoff Between Entropy Rate and Synchronization Reachability

The rate at which entropy blocks are generated imposes fundamental tradeoffs on both the system's responsiveness and its ability to synchronize new devices. Since FSR derives hop decisions from a deterministic entropy generator (MKRAND), synchronization requires the ability to "fast-forward" to a specific entropy block (B_N) based on a known genesis block and a given block height.

Key Observations

- **Fast Entropy Generation (e.g., 1 block per 256 FPGA clock cycles)** enables high-resolution hopping but makes it computationally expensive for new devices to catch up to the current block (e.g., a device would need to compute a few blocks ahead of the current block to predict a future frequency bin for joining).
- **Slower Entropy Generation (e.g., 1 block per 1024 cycles)** reduces the total number of blocks per unit time, enabling new devices to effectively compute ahead from the genesis block, or validate forward-predicted blocks during synchronization.

- **Each block can produce multiple hops**, meaning the entropy generation rate can be significantly lower than the desired hop rate. For instance, with 10-bit frequency bin indices, a single 128-bit block yields up to 12 independent hops.
- **Determinism is preserved regardless of rate**. Slowing the rate of block production does not affect signal unpredictability or mesh coherence, so long as all participants agree on the block production rate and stay within timing tolerances.

Design Implication

To balance operational efficiency and synchronization scalability, FSR recommends a block generation rate tuned such that:

- Each device can produce the current entropy block within a small number of cycles from either the genesis block or a received future sync block.
- The entropy generator (MKRAND) is clocked at a manageable rate (e.g., one block every 1024 FPGA cycles), providing ample time for hardware to perform forward computation.
- The hop scheduler consumes multiple hops per block (e.g., 8–12), allowing the mesh to maintain high-frequency agility without saturating the entropy engine.

Future Optimization

As hardware evolves, adaptive entropy spacing strategies may be introduced: during high-load or jamming scenarios, the mesh may temporarily increase block throughput for higher hop entropy, reverting to a lower baseline rate for routine operation and synchronization tolerance.

5.3 Join Procedure for New Participants

When a new device wishes to join an existing FSR mesh, it faces a key challenge: the current block height may be in the billions or trillions, making it impractical to compute the entropy stream from block zero. To enable scalable and secure onboarding, the following mechanism is used:

1. The new device is seeded with the mesh's shared **genesis block** and a pre-provisioned **pairing key** or **shared sentinel block**.
2. It transmits a **join beacon** within a known rendezvous region (a limited spectrum and timing space derived from the genesis block and reserved for onboarding), encrypted using its pairing key.
3. A nearby node acting as a **sentinel** detects this beacon and responds with:
    - The current or slightly future **block height** (plaintext).

- A corresponding **entropy block** B_{N+K} (encrypted using the shared sentinel block).

4. The joining device decrypts the entropy block, aligns its MKRAND generator to block $N + K$, and immediately begins hopping in sync with the mesh.

5.4 Security Model

- The **genesis block** is never transmitted and must be pre-provisioned.
- The **current entropy block or PSI Index** used for synchronization is always encrypted.
- The **block height** is not sensitive and can be safely broadcast.
- Devices may also use the same pairing infrastructure to exchange encrypted messages, keys, or commands after synchronization.
- Any group of two or more devices who wish to encrypt their communications with each other must be provisioned with a group pairing key during setup.

5.5 Resilience and Statelessness

FSR's synchronization model does not rely on session state or external clocks. Because both sender and receiver derive all timing, frequency, and addressing information from the shared entropy stream, even long periods of disconnection or power loss can be recovered from deterministically. Any node with the correct genesis block and sentinel key can immediately rejoin the mesh.

6 Hashtag-Based Channel Partitioning

To enable logical channel separation and selective listening, each FSR packet includes a 16-bit `tag_mask` field. This field encodes hashtags—predefined labels representing communication topics, roles, or identities.

Receivers configure a bitmask representing the hashtags they are interested in. When a packet is received, its `tag_mask` is bitwise-ANDed with the receiver's listening mask. If any bits match, the receiver processes the packet; otherwise, it is ignored.

This approach provides fine-grained message routing and filtering across a shared spectrum. Tags are consistent across devices sharing the same genesis block, enabling efficient, deterministic communication partitioning in bandwidth-constrained environments.

6.1 Initial Sync Packet Format

Each node maintains a 64-bit **block height** counter, incremented every time a new entropy block is generated. A sync packet consists of:

Tag Bit	Example Hashtag Meaning
0x0001	#status
0x0002	#command
0x0004	#sensor
0x0010	#chat
0x0040	#video
0x8000	#broadcast

Table 1: Sample Hashtag Bitmask Scheme

- **Current Entropy Block or PSI Index** — A 128-bit identifier uniquely representing the current chain state.
- **Block Height** — A 64-bit counter indicating how many blocks have elapsed since the genesis block.
- **Intra-block Offset (optional)** — An integer representing the time (e.g., in microseconds) since the last block was generated.

This packet provides a complete temporal coordinate system, allowing a joining node to align its MKRAND generator and internal clock with millisecond-level precision.

6.2 Embedding Micro-Timing Hints in Data Payload

Since the entropy blocks are generated at fixed clock intervals (e.g., one block per 256 FPGA cycles), nodes can encode the timing of specific events within a block as a simple offset:

- Let block B_N be generated at global time T.
- If an event occurs δt microseconds after T, the transmitting node sends:

$$(\texttt{BlockHash} = B_N,\ \texttt{Height} = N,\ \texttt{Offset} = \delta t)$$

- The receiving node, knowing the same clock rate and block interval, can reconstruct the sender's notion of time with exact resolution.

This scheme avoids floating-point arithmetic and permits accurate time coordination using only integers.

6.3 Drift Correction Method

All nodes in an FSR mesh derive their block schedule from the same entropy sequence and clock rate. However, crystal drift or environmental factors may cause slight deviations over time. To correct for drift:

- Nodes include intra-block timing hints in periodic messages.
- Peers compare the expected block interval against the received offset.
- A simple delta correction algorithm adjusts local timers without modifying block height counters, preserving consistency.

6.4 Re-synchronization Fallback Strategy

If a node drifts too far or misses several blocks, it may fall out of sync. To rejoin the mesh:

1. It reverts to the rendezvous region, where onboarding messages are expected.
2. A sentinel node provides a new encrypted future block and block height.
3. The node resets its MKRAND state and resumes synchronized operation.

This strategy ensures mesh-wide temporal cohesion, even in harsh or mobile environments.

7 Digital Shortcodes as Communication Metadata

All communication is digitally routed and packetized. As such, identity, intent, and routing metadata can be embedded as compact digital shortcodes directly within the data payload. Each FSR burst begins with a small binary header indicating:

FSR Flag Bitfield Definition Example

Bit	Flag Description
0	Encryption enabled (public/private)
1	Acknowledgment required
2	Broadcast (ignore recipient_id)
3	Sync marker (for re-synchronization)
4	Reserved for future use
5–7	Priority level (0–7)

Table 2: Example FSR Header Flag Bits

Example Message Type Codes

Code	Message Type
0x00	Heartbeat / Ping
0x01	Plaintext Message
0x02	Encrypted Message
0x03	Voice Stream Packet
0x04	File Fragment
0x05	Command / Control Signal
0x06	Group Join Request
0x07	Group Invite / Role Assignment
0xFF	Custom / Experimental Payload

Table 3: Example Message Types for FSR Payloads

- **Sender ID** – a unique identifier for the transmitting device
- **Recipient ID** – the intended target, group, or broadcast scope
- **Flags** – control bits for encryption, acknowledgments, group membership, etc.
- **Message Type** – e.g., text, voice, file chunk, command

This enables the formation of workspaces within the radio mesh:

- One-to-one encrypted chats
- Group broadcasts (e.g., "All Medics")
- Role-based coordination (e.g., "Red team to Blue team")
- Hierarchical control signaling

"Digital shortcodes," in this context, become the semantic foundation of FSR's high-level communication layer—providing a minimal yet powerful system for expressive, targeted interaction in a jam-resistant mesh.

8 tag-Based Addressing and Routing

While Full Spectrum Radio (FSR) is primarily designed to provide entropy-driven, jam-resistant communication, the addition of **tag-based addressing** introduces a lightweight yet powerful method of routing messages between multiple participants over a shared frequency hopping pattern.

8.1 Concept of shortcode schemes

Each participant in the network is assigned a unique **shortcode scheme**.

Shortcode schemes are orthogonal to frequency hops: while hopping defines *where* a message is transmitted, the shortcode scheme defines *who* is transmitting and *who* the message is intended for.

8.2 Addressing Modes

FSR supports several communication models via this tag-coding scheme:

- **Unicast:** Specific sender and recipient shortcode schemes are used
- **Broadcast:** Postamble is omitted or set to a reserved wildcard g., "all participants")
- **Groupcast / Anycast:** Short digital codes assigned to roles or teams (e.g., "squad 1," "relay nodes," etc.)

These modes enable a flexible and dynamic routing strategy without the overhead of packet-based headers or encryption schemes.

8.3 Routing Without IP

Shortcode schemes offer a viable alternative to conventional addressing and routing, enabling:

- Stateless message delivery
- Identity recognition
- Natural separation of participants within the same spectral workspace

This architecture allows the creation of peer-to-peer networks, group-based hierarchies, and even public/private channels—ideal for environments where protocol metadata must be minimized or concealed.

"tag-aware FSR" offers a new dimension in decentralized radio communication: one where presence, identity, and intent can be conveyed without ever revealing a packet header.

9 Neighbor Awareness and Local Coordination

In large-scale swarm deployments, especially in GPS-denied or communication-limited environments, autonomous coordination among nearby units is essential. This section outlines mechanisms by which FSR-equipped devices can establish relative awareness of neighbors and implement decentralized behaviors such as flocking, formation control, and collision avoidance.

9.1 Position Awareness Without GPS

Several techniques can be employed by devices to estimate proximity or bearing to neighboring units, without requiring external infrastructure:

- **RSSI (Received Signal Strength Indicator):** Devices periodically emit low-power `#ping` packets. Neighbors record the received signal strength and infer proximity. While imprecise, RSSI provides a low-cost and hardware-free method for range estimation.
- **Time-of-Flight (ToF):** Using wideband FSR pulses or UWB extensions, devices can estimate round-trip travel time of signals. This enables more accurate distance measurements (within 10–50 cm) using commodity UWB chipsets.
- **Angle-of-Arrival (AoA):** Devices equipped with two or more antennas can estimate the direction from which a signal was received. Combined with ToF or RSSI, this allows construction of a local polar coordinate map of neighboring units.
- **Infrared or Ultrasonic Line-of-Sight:** For dense indoor environments, devices may use simple IR or ultrasonic pings to detect and track neighboring units with high spatial precision.

- **Hybrid FSR + BLE Scanning:** Devices equipped with Bluetooth Low Energy (BLE) can perform passive scans while using FSR for primary communication. BLE advertisements provide short-range identity and signal strength cues to help populate a local neighbor table.

9.2 Decentralized Flocking Behavior

Using local neighbor observations, devices can execute distributed algorithms such as Reynolds-style flocking, without base station involvement:

- **Separation:** Move away from units with overly strong RSSI or small ToF readings.
- **Alignment:** Share and align heading vectors with nearby units via tagged `#heading` messages.
- **Cohesion:** Move toward weaker neighbor signals (more distant members of the same swarm group).

These behaviors enable robust, scalable movement patterns without reliance on GPS, centralized control, or dedicated inter-unit links.

9.3 Security and Coordination Constraints

All inter-unit messages are constrained by the swarm's shared entropy source (i.e., genesis block), ensuring only authenticated members participate in proximity coordination. Messages are tagged (e.g., `#localcoord`, `#heading`) and routed efficiently using FSR's tag-based addressing scheme, minimizing spectrum overhead.

10 Secure Messaging over FSR: Public and Private Communication Channels

Full Spectrum Radio (FSR) inherently provides confidentiality against outside observers due to its entropy-driven frequency hopping, which renders interception impractical without shared initialization parameters. However, within a shared network—where multiple devices operate using the same genesis block and hop sequence—further cryptographic measures are needed to enforce **message privacy, authenticity, and recipient specificity**.

10.1 Transmission Modes

1. Public Broadcast Mode

In this mode, messages are transmitted over the shared FSR hopstream in an unencrypted form. All devices synchronized to the hop sequence (i.e., sharing the same genesis block) can receive and interpret the message. Although the payload is not encrypted, it remains inaccessible to adversaries without synchronization.

- **Use Case:** Open group communication, environmental telemetry, or non-sensitive coordination.
- **Security:** Encrypted by obscurity — only devices on the correct hop pattern can receive it.

2. Private Encrypted Mode

For confidential exchanges, the sender encrypts the message payload using the recipient's shared pairing block. The message is still broadcast on the shared hopstream, but only the intended recipient—who holds the shared pairing block—can decrypt and access its contents.

- **Use Case:** Private chat, command-and-control signaling, secure telemetry.
- **Security:** Payload is end-to-end encrypted using shared private-key cryptography.

11 Private Communication via MKRAND Pairing

While Full Spectrum Radio (FSR) enables group-wide communication using a shared entropy genesis block, there are scenarios in which point-to-point confidentiality is required. Rather than relying on conventional math-based asymmetric encryption (e.g., RSA or ECC), FSR leverages MKRAND's entropy blocks as symmetric keys, enabling private channels between previously paired devices without introducing cryptographic overhead or third-party trust.

11.1 Pairing Model and Key Sharing

MKRAND blocks are fundamentally symmetric: an entropy block used to encrypt a message can also be used to decrypt it. To facilitate private communication, two devices must first undergo a secure pairing process, during which they exchange or are provisioned with a shared entropy block unique to their relationship. This block, referred to as the *pairing block*, is stored in secure local memory and used exclusively for communication between the paired devices.

Once this block is shared:

- Either device can use it to XOR-encrypt arbitrary-length plaintext.
- The recipient decrypts the message using the same block and XOR logic.
- Devices that are not part of the pairing cannot decrypt or infer any meaningful data, even if they share the broader genesis block.

11.2 Dual-Channel Communication Capability

Devices operating in an FSR network can engage in two types of transmissions:

- **Public**: Messages encrypted using the network's shared *genesis block*, readable by all devices within the group.

- **Private**: Messages encrypted using a pairing block, readable only by the intended recipient with whom the block is shared.

This dual-channel model allows devices to communicate freely across the network while retaining the option to engage in private, tamper-proof conversations.

11.3 Example Scenario

Consider devices A, B, and C:

- All share a common genesis block `block_G`, allowing them to receive public broadcasts.
- Devices A and B also share a pairing block `block_AB`.

Device A can then:

- Send a general message using `block_G` — all nodes receive and decode it.
- Send a private message using `block_AB` — only B can decrypt it.
- Device C hears both transmissions but can only decode the one using `block_G`.

11.4 Security Properties and Limitations

- **No PKI Required:** Encryption and decryption use the same block, eliminating the need for public/private key infrastructure.
- **Spoofing Resistance:** Adversaries cannot spoof participants unless they possess the pairing block.
- **Replay Mitigation:** If blocks are consumed linearly, repeated messages yield different ciphertexts.
- **Controlled Pairing:** Pairing must occur through a secure channel — via hardware, QR scan, or NFC — and cannot be dynamically negotiated over the air.

11.5 Future Enhancements

- **Time-Based Pairing Blocks:** Use MKRAND-derived time-indexed entropy to rotate pairing blocks.
- **Hierarchical Groups:** Allow for squad-, platoon-, or team-level blocks layered atop the genesis block.
- **Key Revocation:** Implement a method for securely erasing or invalidating old pairing blocks from local memory.

This model preserves the high-speed, hardware-driven nature of FSR while enabling secure, scalable communication patterns suitable for mesh networks, battlefield deployments, or swarm robotics.

11.6 Security Implications

- **Eavesdropping by outsiders:** Effectively impossible due to the entropy-based hopstream.
- **Eavesdropping by insiders:** Mitigated by encrypting payloads with shared private keys, or pairing blocks.
- **Spoofing:** Only feasible by insiders with authorized equipment; mitigated by signed metadata or trust overlays.

In combination, these features elevate FSR from a novel physical layer protocol into a robust framework for secure, multi-party, real-time communication with privacy controls and participant authentication.

A FSR Protocol Framework

The Full Spectrum Radio (FSR) protocol is designed to operate as a lightweight, digitally native communication standard tailored for entropy-driven frequency-hopping systems. While the foundational concepts—such as deterministic entropy synchronization, secure mesh joining, and digitally timed communication—are fixed, the precise binary encoding, message formats, and field definitions are implementation-specific and are expected to evolve during engineering integration.

A.1 Deferred Field Specifications

Rather than prescribing a fixed packet layout at this stage, FSR defines a flexible architecture wherein the following categories of data are expected to be encoded at the protocol layer:

- **Header Flags** — Boolean and multibit fields that may specify encryption mode, priority level, broadcast targeting, or stream type.
- **Message Type Codes** — Numeric opcodes used to distinguish payload intent (e.g., plaintext, encrypted, sync, join request, key advertisement, etc.).
- **Timing Metadata** — Fields to encode block height, intra-block time offset, and drift correction parameters.
- **Encryption Metadata** — Optional fields used to specify or derive symmetric key references, sentinel block usage, or pairing identifiers.
- **Payload Section** — Application data, sensor readings, control messages, or compressed stream segments.

The exact bit widths, byte orderings, and encoding strategies will depend on system constraints, including available bandwidth, microcontroller architecture, and security policy. Implementers are encouraged to optimize protocol efficiency and extensibility based on deployment context.

Note: Protocol-level interoperability should be guided by open schema definitions (e.g., CBOR, Protocol Buffers, or custom binary schemas) that reflect the intent of the fields outlined above, while preserving the architectural principles laid out in this document.

B Scalability and ID Space

The Full-Spexctrum Radio (FSR) protocol is designed with scalability in mind, enabling flexible deployment across small IoT swarms, large-scale industrial fleets, and military-grade autonomous systems.

B.1 Participant Capacity

A 12-bit `device_id` supports up to 4096 participants per genesis block, exceeding the scale of currently deployed coordinated drone swarms. This accommodates:

- Multi-drone orchestration (e.g., light shows, search-and-rescue, **combat-ready autonomous swarms for ISR and precision strike**)
- Ground-robot squads with mixed capabilities
- Sensor nets or mobile data collection nodes

For future-proof deployments, the protocol can optionally extend `device_id` and `tag_id` fields to 16 bits, supporting 65,536 participants and hashtags respectively.

B.2 Hashtag Routing Model

The `tag_id` acts as a routing hint. Devices can subscribe to one or more tag IDs corresponding to their role (e.g., `#targeting`, `#logistics`, `#sensorstream`). This allows high-bandwidth operations without requiring full device-to-device addressing.

B.3 Security Implications

Large ID spaces also mitigate spoofing. Since devices are deterministically seeded via a shared genesis block, unauthorized participants cannot generate valid entropy blocks or hop patterns, making impersonation infeasible.

B.4 Design Considerations

- Bitfields may be adjusted per deployment profile.
- Tag allocation can be pre-coordinated or dynamically negotiated using control messages.
- Devices can ignore irrelevant traffic by filtering on `tag_id`, improving efficiency.

B.5 Behavioral Semantics

- Sentinel devices continuously monitor the rendezvous spectrum during idle windows for onboarding requests.
- Messages with the `broadcast` flag are routed to all peers sharing the genesis block.
- When `encryption` is enabled, the payload must be XOR'd with a pre-shared secret or target public key stream.
- Sync markers enable catch-up or initialization behaviors in newly listening devices.
- Critical messages are prioritized in transmission queues and may override typical collision-handling backoffs.

Patent Claims

1. A wireless communication protocol based on deterministic entropy streams derived from a shared genesis block, enabling frequency hopping without pseudorandom number generators.
2. A method for synchronizing wireless mesh nodes via a shared entropy generator (MKRAND), seeded by a genesis block, wherein identical entropy outputs are deterministically produced across devices.
3. A secure onboarding procedure whereby a sentinel node transmits a future entropy block encrypted with a pre-provisioned key to enable joiners to align their internal entropy state with the mesh.
4. A timing mechanism based on block height, where each block represents a fixed number of clock cycles, enabling relative timing and synchronization without external clocks.
5. A frequency-hopping system where hopping bins are derived from segments of a 128-bit entropy block, enabling multiple hops per block and spectrum agility.
6. A rendezvous region composed of reserved frequency bins and entropy block ranges, enabling new devices to broadcast join messages without requiring global synchronization.
7. A synchronization fallback strategy that allows devices to recover alignment using any valid encrypted message from a sentinel node without needing explicit SYNC frames.
8. A modular block-based mesh protocol that allows rollover of block height without loss of synchronization, supporting long-lived networks exceeding thousands of years of uptime.
9. The use of micro-timing hints embedded in data payloads to support drift correction and intra-block synchronization refinement.

10. Support for decentralized swarming behavior, including drone orchestration, through shared entropy streams and synchronized hopping, without centralized time coordination, and with optional integration of position awareness using radio-based techniques such as Received Signal Strength Indicator (RSSI), Time-of-Flight (ToF), and Angle-of-Arrival (AoA) for dynamic spatial coordination.