HekateForge Shared Entropy Pool Choreography (SEPC) Protocol

A Cryptographic Paradigm for Secure, Deniable, and Forensically Resistant Communication

By: **William Appleton**

Patent Pending, Application #3278624

Abstract

This whitepaper introduces the HekateForge Shared Entropy Pool Choreography (SEPC) Protocol, a novel communication framework that replaces conventional encrypted transmission with cryptographic referencing of shared entropy. No ciphertext ever traverses a network, instead, plaintext is re-constructed through ephemeral coordinate mappings into a shared entropy space. By leveraging time-bound entropy pools and deterministic XOR-based segment encoding, the protocol achieves bandwidth-efficient, deniable communication suitable for quantum-secure applications.

1. Motivation

Standard encryption schemes rely on transmitting ciphertext, which - while undecipherable without keys - is still detectable, inspectable, and potentially vulnerable to future decryption. EPCP redefines this boundary by never transmitting the encrypted message content at all. Instead, it uses shared references into public entropy to encode and decode messages in an entirely choreography-based fashion.

2. Core Components

2.1 Entropy Pools

- 25KB segments of randomized ASCII
- Base64-encoded for transport consistency
- Stored in publicly accessible infrastructure (e.g., Apache directory)
- Include metadata:
    - PoolID – unique identifier
    - TTL – Unix timestamp defining 5-minute validity
    - GeneratedAt – timestamp of entropy creation
    - VerificationHash – SHA-256 hash of raw Data
    - Data – 25KB ASCII segment containing all required Base64 characters.

Example JSON:


```
{
  "PoolID": "abc123",
  "TTL": 1751918922,
  "GeneratedAt": 1751918622,
  "VerificationHash": "89db0e6d825aac0ec59f7469b21121bb3f60a8ef95f7e3c349e5f3bfa…",
  "Data": "xzZgQ8dGDM3xCTBNdLcV4ECKPbpDmIMmBfjKqEeM9bW=HVLqCRqAxBVn/3YSNn…"
}
```

2.2 Key Exchange Layer

- Quantum Key Distribution (QKD):
    - Preferred method
    - Transmits PoolID, segment length, Seed, and LIC securely
- Public Key Infrastructure (PKI):
    - Fallback for classical environments
    - Offers computational-level security

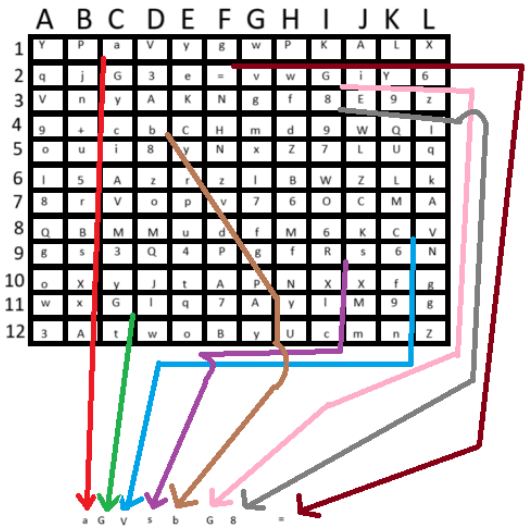
2.3 HekateForge Encoding Layer

- Shared entropy pool is digested.
- Plaintext message is converted to Base64
- The byte values for the Base64 string are mapped to the entropy pool digest.
- The index values resulting from the previous step are encoded into a non-delimited string of integers.
- The string resulting of the last step is chunked into segments of X length.
- The chunks are XORed with keys whereby:
    - Key#1 = SUM(LIC_Bytes)*Seed
    - CipherText#1 = PlainTextChunk#1 XOR Key#1
    - Key#2 = SUM(PlainTextChunk#1_Bytes)*(1*SegmentLength)
    - CipherText#2 = PlainTextChunk#2 XOR Key#2
    - Key#3 = SUM(PlainTextChunk#2_Bytes)*(2*SegmentLength)
    - ….Until all chunks are accounted for.
- The values resulting from the previous point are joined into a string separated with a non-numerical ASCII.
- The string resulting from the previous point is compressed and transmitted (PLD).

2.4 Message Reconstruction Visual

HekateForge Entropy Pool Message Reconstruction

PLD Message
1C, 11C, 8L, 9J, 4D, 2I, 3I, 2F



Decoded Base64 Message
aGVsbG8=
Plaintext Message
hello

3. Security Properties

- No Ciphertext in Transit: No payload to intercept or inspect
- Perfect Forward Secrecy: Entropy pools expire every 5 minutes rendering even a decoded PLD useless.
- Forensic Resistance: The PLD cannot reconstruct message without shared entropy
- Quantum Readiness: Compatible with both QKD and post-quantum PKI

4. Applications

- Military / Government Communications
- Medical Device Sync
- SCADA System Communication
- Secure Corporate Messaging