StegoCrypt — mathematical description

1. Goals & assumptions

Goals. Hide a secret payload (bytes) inside a cover image or video such that:

- The hidden bytes are **confidential** (requires password to decrypt).
- The modifications are **visually improbable** and **statistically difficult** to detect by casual inspection.
- The embedding/extraction is **deterministic** given the correct password (for reproducibility).
- Optionally, add **error-correction** for robustness to limited corruption.

Threat model.

- Passive adversary: can inspect the stego file, compare with known covers, or run automated steganalysis.
- Active adversary: may re-encode or compress (lossy) the file.
- StegoCrypt assumes sender and receiver can exchange the password securely out of band.

2. Payload packaging & cryptography

2.1 Payload format

Let the secret payload be a byte string M. We build an encapsulated payload P:

1. Salt: $s \leftarrow \text{Random}(\ell_s)$(e.g., 16 bytes)
2. Key derivation: $K = \text{KDF}(\text{password}, s)$. StegoCrypt uses PBKDF2 (HMAC-SHA256) with iterations $t$and salt s.
3. AES-GCM encryption: choose random nonce n(e.g., 12 bytes), compute $C, T = \text{AES\_GCM\_Encrypt}(K, M; n)$, where $T$is the authentication tag.
4. Header: fixed magic bytes $H$and payload length $L = \mid C \mid + \mid T \mid + \mid n \mid$encoded as fixed-size integer.

5. Optional ECC: if Reed–Solomon (RS) is enabled, apply RS encoding to the ciphertext+tag to produce E.

Final bytes to embed:

$$P = H \parallel L \parallel s \parallel n \parallel C \parallel T \text{ (or RS-coded version)}$$

Notes:

- AES-GCM provides confidentiality + integrity. On extraction, tag verification fails if either password wrong or ciphertext corrupted → "MAC check failed".
- KDF parameters (salt size ℓ_s, iterations t) control brute-force resistance.

2.2 KDF formula

Use PBKDF2 with HMAC-SHA256:

$$K = \text{PBKDF2}_{\text{HMAC-SHA256}}(\text{password}, s, t, k)$$

where $k$is the desired key length (e.g., 256 bits). Recommended $t \geq 100{,}000$(adjust for CPU).

3. Bit mapping, LSB embedding and capacity

3.1 Cover representation

- Images: each frame is an array of pixels with $c$color channels (typically $c = 3$for RGB). A pixel channel is a byte $b \in \{0, \ldots, 255\}$.
- Videos: sequence of Fframes, each frame of resolution $W \times H$. Total pixel channels:

$$N_{\text{slots}} = F \cdot W \cdot H \cdot c$$

Each slot holds up to Bbits if using Bleast significant bits (LSBs) per channel.

3.2 Capacity

Number of embeddable bits:

$$\text{Capacity\_bits} = N_{\text{slots}} \cdot B$$

Number of embeddable bytes (payload capacity):

$$\text{Capacity\_bytes} = \lfloor \frac{\text{Capacity\_bits}}{8} \rfloor$$

Example. 320×240 video, 100 frames, $c = 3$, $B = 1$:

$$N_{\text{slots}} = 100 \cdot 320 \cdot 240 \cdot 3 = 23{,}040{,}000$$

Capacity bytes ≈ 2.88 MB.

3.3 LSB embedding rule (per-slot)

For each payload bit $x \in \{0,1\}$to embed into a channel byte $b$and using BLSBs:

- Let $b_{\text{msb}} = b\& \sim ((1 \ll B) - 1)$(clear the low Bbits).
- Let $b' = b_{\text{msb}} \mid x$(if $B = 1$) or $b' = b_{\text{msb}} \mid$ next B payload bits.
- Write b'back.

This changes channel value by at most $2^B - 1$. With $B = 1$, $| b' - b | \in \{0,1\}$— minimal perceptual difference.

4. Spreading (pseudo-random permutation) for stealth

4.1 Rationale

Sequentially writing payload bits in natural raster order produces visible/statistical artifacts (clusters, edges). Spreading pseudo-randomly distributes payload bits across slots to:

- Avoid contiguous clusters
- Reduce local statistical anomalies
- Make statistical detection harder

4.2 Deterministic permutation

Let $S = \{0, \dots, N_{\text{slots}} - 1\}$be the ordered list of available slots (channel indices across frames). We compute a deterministic permutation $\pi$of Sderived from the password and salt so both embedder and extractor reproduce it.

Common construction:

- Use a PRNG (cryptographic stream cipher or CSPRNG) seeded with $seed = \mathrm{HMAC}(K_{\text{perm}}, s)$where $K_{\text{perm}}$is derived from the password (or reuse K).
- Generate a random permutation via Fisher–Yates using the PRNG:
- A = [0..N-1]
- for i = N-1 down to 1:
- j = PRNG.next_integer(0..i)
- swap(A[i], A[j])
- Permutation $\pi(i) = A[i]$.

Because the header and salt are embedded sequentially at known low offsets (so the extractor can read salt/len early), the same permutation is reproducible.

4.3 Per-chunk permutation for streaming

For memory efficiency in video streaming:

- Process frames in chunks of Cframes. For chunk k, define $N_k = C \cdot W \cdot H \cdot c$.
- Use seed derived from $seed_k = \mathrm{HMAC}(K_{\text{perm}}, s \parallel k)$so each chunk uses a deterministic permutation π_k.
- This allows embedding/extraction chunk-by-chunk without permuting the entire video at once.

5. Reed–Solomon ECC (optional)

5.1 Purpose & parameters

Reed–Solomon (RS) adds parity symbols to correct bit/byte corruptions. In StegoCrypt it operates on payload bytes after encryption but before embedding.

RS parameters: RS(n, k) over GF(2^8) where:

- message length $k$bytes,
- codeword length $n = k + 2t$,
- correctable symbol errors $t = \frac{n-k}{2}$.

If you set rs_nsym = 32, you add 32 parity bytes → correct up to 16 byte errors.

5.2 Trade-off

- ECC increases the number of bytes to embed by factor $\frac{n}{k}$.
- Larger ECC → better robustness to corruption but larger stego payload → more capacity required and potentially larger detectability footprint.

5.3 Integration

Pipeline:

$$M \overset{\text{encrypt}}{\rightarrow} C \overset{\text{RS encode}}{\rightarrow} E \overset{\text{embed}}{\rightarrow} \text{stego}$$

On retrieval: extract bits → bytes → RS decode → AES-GCM decrypt → verify tag.

If RS decoding fails or AES-GCM tag fails, extraction aborts (integrity failure).

6. Video-specific considerations (streaming, codecs)

6.1 Streaming chunking

Frames are processed in chunks C. For chunk $k$the mapping of global slot index to (frame, y, x, channel) is:

$$\text{slot\_index} \mapsto (f = \lfloor \frac{\text{slot\_index}}{W \cdot H \cdot c} \rfloor,\ y = \lfloor (\text{slot\_index}\ mod\ (W \cdot H \cdot c))/(W \cdot c) \rfloor,\ x = \lfloor ((\text{slot\_index}\ mod\ (W \cdot H \cdot c))\ mod\ (W \cdot c))/c \rfloor,\ ch = (\cdot))$$

When using per-chunk permutations, reuse the chunk-local seed.

6.2 Lossless vs lossy codecs

- Lossless codecs (FFV1, libx264rgb lossless mode) preserve decoded pixel values exactly → ideal for LSB steganography.
- Lossy codecs (standard H.264 with chroma subsampling, motion compression) will change LSBs unpredictably → will usually break simple LSB extraction unless heavy ECC and redundancy are used.

7. Steganalysis & detectability

7.1 Simple statistical tests

- **LSB plane analysis.** For natural images the LSBs behave close to random, but certain patterns differ by region. Embedding uniformly can increase entropy or break expected correlations.
- **Chi-square test** (or RS analysis) can detect changes to LSB distributions.
- **Sample pairs analysis** looks for deviations in parity frequency.

7.2 Effect of spreading

Permutation spreading reduces local cluster changes, making spatial statistics more natural; however global statistics (histograms, pairwise correlations) may still change.

7.3 Metrics

- Let $\Delta(b)$denote distribution of bit flips across intensities or local neighborhoods. A strong stealth metric is minimizing the KL divergence between the empirical LSB distribution of cover and stego:

$$D_{\mathrm{KL}}(P_{\text{cover}} \parallel P_{\text{stego}}) = \sum_{x} \quad P_{\text{cover}}(x) \log \frac{P_{\text{cover}}(x)}{P_{\text{stego}}(x)}.$$

Spreading + B=1 minimizes detectable divergence vs larger B.

7.4 Practical recommendation

- Use $B = 1$and spread (default) for best stealth.
- Avoid embedding near image edges or uniform large flat regions (but spreading already mitigates this).
- Do not re-encode with lossy codecs after embedding.

8. Error probability & reliability (back-of-envelope)

Assume an adversary or transmission causes independent per-channel bit-flip probability p. For $B = 1$and payload of $m$bytes, expected number of corrupted bytes depends on bit-to-byte mapping and deinterleaving (spreading acts as interleaver).

If RS corrects up to $t$byte errors, probability of successful decode is:

$$P(\text{success}) = \sum_{e=0}^{t} \binom{m + n_{\text{par}}}{e} q^{e} (1-q)^{(m+n_{\text{par}})-e}$$

where $q$is the byte-error probability (derived from bit flip model). This is approximative.

9. Algorithms & pseudocode

9.1 Embed (high-level)

Input: cover (image/video frames), password, message bytes M, params: B, use_spread, use_rs, rs_nsym

1. P = build_payload(M, password, use_rs=use_rs, rs_nsym=rs_nsym)

2. header = H || len(P) || salt

3. bits = bytes_to_bitstream(header || P)

4. if use_spread:

    compute permutation(s) pi (global or per-chunk) seeded by password+salt

    slots = apply permutation(pi) # an ordered list of channel indices to write into

  else:

    slots = [0..N_slots-1]

5. for i, bit in enumerate(bits):

    slot = slots[i]

    write_bit_into_slot(slot, bit, B)

6. save stego cover (frames->container). Use lossless codec.

9.2 Extract (high-level)

Input: stego file, password, params B, use_spread, use_rs

1. Read first N_header_slots sequentially to recover header(H, len, salt)

2. derive K, permutation seed from password+salt

3. build slots via permutation (same as embed)

4. read bitstream by reading low-B bits from slots

5. reconstruct bytes, optionally RS-decode

6. attempt AES-GCM decrypt with K and nonce; if tag valid, output M

10. Security analysis

10.1 Confidentiality

- AES-GCM with strong KDF (PBKDF2 with sufficient iterations) gives standard IND-CPA/AE guarantees: without password, ciphertext indistinguishable from random (subject to nonce reuse rules).
- Salt and nonce must be random and unique per embedding.

10.2 Integrity

- AES-GCM tag prevents undetected modification. If tag verification fails extraction reports failure; RS cannot correct arbitrary malicious tampering beyond its correction capability.

10.3 Stealth

- Confidentiality does not imply stealth: ciphertext has high entropy and if embedded densely may increase local entropy and be detectable.
- Spreading reduces local detectability; limiting per-channel modification to 1 LSB is the primary practical stealth tradeoff.

10.4 Brute-force & password strength

- Attacker can attempt password guesses. Each attempt requires full extraction and AES verification. Use high KDF work factor and encourage strong passwords.

11. Parameter recommendations (practical)

- LSB $B = 1$(default) — best stealth.
- Use spread = True.
- RS only if you anticipate lossy transforms. If used: rs_nsym set based on expected corruption; 32 parity bytes correct up to 16 byte-errors.
- KDF: PBKDF2-SHA256 with iterations $t = 200{,}000$(adjust to CPU), salt 16 bytes.

- AES-GCM: 256-bit key, 12-byte nonce, 16-byte tag.
- Chunk size for video: $C = 30$–90 frames, depending on memory.

12. Example numeric scenario

320×240, 60 frames, $c = 3$, $B = 1$:

- slots: $60 \cdot 320 \cdot 240 \cdot 3 = 13{,}824{,}000$bits → capacity ≈ 1.73 MB payload bytes.
- After AES-GCM + header + salt (assume 128 bytes overhead), usable payload ≈ 1.73 MB − 128 B.