CRISP: Channel-Randomised Single-Image Steganography with Permutations

Shahzad Ahmad[1 [0000−0002−9654−869X]] and Stefan Rass[1 [0000−0003−2821−2489]]

LIT Secure and Correct Systems Lab, Johannes Kepler University Linz,
Altenbergerstrasse 69, 4040 Linz, Austria
shahzad.ahmad@jku.at, stefan.rass@jku.at

Extended version. This is the full version of the paper appearing at the 28th International Conference on Information and Communications Security (ICICS 2026), Fukui, Japan, Springer LNCS. It contains the complete proof of Theorem 2, the case enumeration behind Proposition 3, the full experimental tables, the correctness proof, and the empirical steganalysis, which the conference version abbreviates for space. The technical claims are identical in both versions.

Abstract. We introduce CRISP (Channel-Randomised Single-Image Steganography with Permutations), a homomorphic steganography scheme for outsourced computation. In the setting we consider, a client (Alice) hides Boolean inputs in the least-significant bits of cover images and asks an honest-but-curious cloud (Carol) to evaluate a logic circuit, gate by gate, directly on those images so that a receiver (Bob) can later extract the result. The setting is natural for outsourcing but non-standard for steganography: Carol knows that steganographic embedding is used, knows the scheme, and knows the public channel-assignment permutations; the only secret is the per-execution pixel position (row, col) in the image at which the bits live. The security goal is therefore positional hiding under known presence, not Cachin-style undetectability.

CRISP embeds all three inputs of a Fredkin gate (a universal reversible three-bit logic gate) into the three RGB channels of a single cover image at a secret pixel position, and writes all three outputs into a single output cover at the same position. Two independently sampled permutations $(\pi_{\text{in}}, \pi_{\text{out}}) \in S_3 \times S_3$ assign channels to logical roles at the input and output of each gate, and both travel with the public circuit specification. Two results about the limits of this design follow, and we regard them as the more useful contribution. First, per-gate resampling of π_{out} does not give circuit privacy. We prove that a server holding the circuit specification reads the channel-to-role map at every gate directly, and that a weaker adversary holding only the images recovers the same map and the wiring graph with nine channel-pair comparisons per gate. Second, a $1/(h{\times}w)$ positional bound proved on a single image does not survive a full multi-image transcript when ancillary wires carry publicly known constants. We restate the security game over the whole transcript and prove a bound $2^{\lambda}/(2^{\lambda} + n - 1)$ with $n = h{\times}w$ and λ the gap between the number of secret input bits and the collision entropy of the server's prior on them. Constants embedded only at the secret pixel push λ up by

one bit each, and enough of them pin the pixel down exactly. Two cheap repairs drive λ back to zero and restore the exact $1/(h\times w)$ bound. The decay is polynomial, not super-polynomial, so the bound is statistically small but not cryptographically negligible in the standard sense.

Keywords: Homomorphic steganography · Fredkin gate · Privacy-preserving computation · Positional security · Cloud computing · Reversible logic

1 Introduction

Outsourcing computation to a third-party cloud is cheap and scalable, and it raises one hard problem: letting an untrusted server compute on data without telling it anything about the data. Fully Homomorphic Encryption [17] solves this in generality at an overhead that is still prohibitive at scale. Trusted Execution Environments [11] solve it in hardware, at the price of specific CPU features, attestation, and a side-channel record [5]. Traditional steganography hides that a message exists but offers no way to compute on it.

Provably Secure Homomorphic Steganography (ProSt) [1] takes a third route: hide secret bits in the least-significant bits of cover images and let the server evaluate Fredkin gates [15] over every pixel, so it processes the secret without knowing which pixel holds it. The adversary's advantage falls as the number of candidate positions grows.

What CRISP is and is not. The threat model differs fundamentally from classical undetectability-based steganography [7,16], and that distinction drives every design choice in CRISP. The cloud (Carol) knows the scheme. She knows steganographic embedding is in use, knows the cover format, and possesses the full circuit specification, including the channel-assignment permutations π_{in} and π_{out}. These permutations describe how the RGB channels are mapped to the three logical input roles (and output roles) of each Fredkin gate; they do not describe the circuit's wiring, which is also public. The security goal is therefore not to hide the existence of stego content from her, an aim that is incompatible with outsourced computation by definition, but to hide *which pixel* carries the secret. That is the whole of it. Section 6.2 shows that the channel-to-role map at each gate is not hidden and cannot be, so positional uncertainty is the only thing playing the role traditionally filled by an encryption key. We make this model explicit in Section 4.

ProSt faces a clear limitation within this model: each circuit wire needs a dedicated cover image. A circuit with 61 wires requires 61 images, creating significant communication and storage overhead. ProSt also leaks the wire-to-role map, since wire roles are tied to specific images. CRISP was designed to fix both. It fixes the first and, as Section 6.2 shows, it does not fix the second.

Our Contributions. This paper contributes:

1. **Single-image Fredkin embedding.** All three Fredkin inputs are embedded into the three RGB channels of one input cover at a secret pixel (row, col);

all three outputs are written to one independently sampled output cover at the same pixel. Each gate consumes 2 images (vs. 6 in ProSt), giving a 3× per-gate image-count reduction and a circuit-level reduction of $\approx L/(2n)$.

2. **A negative result on permutation-based circuit privacy.** Resampling π_{out} per gate hides nothing. A server holding the circuit specification inverts the permutations directly (Proposition 2). An adversary holding only the image set recovers the per-gate control channel and the full wiring graph with nine channel-pair comparisons per gate at $O(hw)$ bit operations each (Proposition 3). We state this plainly because the flawed argument is intuitive and other steganographic-computation designs are likely to reach for the same trick.
3. **A transcript-level security bound.** We restate the security game over the full multi-image transcript rather than a single free triple, and prove $\mathrm{Adv}^{\mathsf{joint}} = \mathbb{E}[1/(1+B)] \geq 2^{\lambda}/(2^{\lambda}+n-1)$, where $B \sim \mathrm{Bin}(n-1, 2^{-\lambda})$ and $\lambda = m - H_2(q)$ measures the gap between the m secret input bits and the collision entropy of Carol's prior q on them (Theorem 3). The earlier $1/n$ claim is the special case $\lambda = 0$.
4. **Two repairs that restore $\lambda = 0$.** Publicly known constant wires embedded only at the secret pixel add one bit to λ each, and enough of them locate the pixel exactly (Corollary 1). Giving such wires pixel-uniform LSB planes removes the leak and removes their carriers from the transmission. Re-randomising the LSB plane of each secret carrier turns the i.i.d. cover assumption from a modelling convenience into a property of the scheme.

The remainder of this paper is organised as follows. Section 2 reviews related work. Section 3 provides background on LSB steganography, Fredkin gates, and ProSt. Section 4 states the threat model. Section 5 presents the CRISP scheme. Section 6 provides the security analysis. Section 7 describes the implementation. Section 8 presents experimental results. Section 9 concludes.

2 Related Work

Steganography. LSB substitution [8], pixel-value differencing [22], adaptive methods [20] and deep-learning encoders [2, 25] all treat steganography as pure hiding, with no way to compute on the hidden data. The classical security model targets undetectability, formalised by Cachin [6,7]. Recent provably secure schemes [13, 24] achieve strong undetectability under specific channel assumptions but, again, do not enable homomorphic computation. CRISP operates in a different setting: the cloud knows steganography is in use, and security comes from positional and statistical hiding under that assumption.

Homomorphic encryption. Gentry's FHE [17] and its successors [4, 9, 10, 14] compute over ciphertexts but pay in bootstrapping, key management and ciphertext expansion. CRISP uses no encryption.

TEEs and MPC. Intel SGX [11], ARM TrustZone and AMD SEV provide hardware isolation but need specific CPU features, require attestation, and carry

a documented side-channel record [5,19]. MPC protocols [12,18,23] need multiple interaction rounds, which asynchronous cloud outsourcing does not offer.

Circuit privacy and ProSt. Garbled-circuit circuit privacy [3,23] hides the circuit beyond its input-output behaviour, and private function evaluation reaches the same goal by evaluating a universal circuit of size $\Theta(|C|\log|C|)$ [21]. Neither notion is achieved here, and Section 6.2 shows that no weaker per-channel analogue is achieved either. ProSt [1] introduced homomorphic steganography via Fredkin gates with one cover image per circuit wire; its image count is what CRISP improves.

3 Background

3.1 LSB Steganography

An image I is a three-dimensional tensor $I \in \mathbb{Z}_{256}^{h\times w\times 3}$, where h, w are height and width in pixels and the third dimension indexes channels $\{0,1,2\} = \{R,G,B\}$.

Definition 1 (LSB Embedding and Extraction). *For image I, secret bit $b \in \{0,1\}$, and position (row, col, k):*

$$\mathrm{Emb}(I, b, row, col, k) : I[row, col, k] \leftarrow (I[row, col, k] \;\&\; \neg 1) \mid b \tag{1}$$

$$\mathrm{Ext}(I, row, col, k) : b \leftarrow I[row, col, k] \;\&\; 1 \tag{2}$$

The modification changes pixel intensity by at most 1. Security rests on positional secrecy rather than imperceptibility: without (row, col) an adversary cannot say which of the $h \times w$ pixels carries the secret.

3.2 The Fredkin Gate

The Fredkin gate $F : \{0,1\}^3 \to \{0,1\}^3$ is a universal reversible logic gate defined as:

$$F(c,x,y) = \begin{cases} (c,x,y) & \text{if } c = 0 \\ (c,y,x) & \text{if } c = 1 \end{cases} \tag{3}$$

where c is the control bit and x, y are data bits. Two properties are central to CRISP.

Lemma 1 (Universality [15]). *The Fredkin gate can simulate AND, OR, and NOT:*

$$\mathrm{AND}(a,b) = F(a,b,0)_y \qquad \mathrm{NOT}(x) = F(x,1,0)_x \qquad \mathrm{OR}(a,b) = F(a,1,b)_y$$

where subscripts denote output wire selection. Any Boolean function can be expressed as a network of Fredkin gates.

Lemma 2 (Hamming Weight Invariance [1]). *For any $(c,x,y) \in \{0,1\}^3$, the Hamming weight satisfies $c+x+y = c_{\text{out}}+x_{\text{out}}+y_{\text{out}}$, where $(c_{\text{out}}, x_{\text{out}}, y_{\text{out}}) = F(c,x,y)$.*

3.3 ProSt Summary

ProSt [1] assigns one cover image per circuit wire. For each gate G_i with inputs (I_c, I_1, I_2) and outputs (O_c, O_1, O_2), the cloud applies:

$$\big(\mathrm{lsb}(O_c),\, \mathrm{lsb}(O_1),\, \mathrm{lsb}(O_2)\big)[r, c, k] \;\leftarrow\; F\big(\mathrm{lsb}(I_c),\, \mathrm{lsb}(I_1),\, \mathrm{lsb}(I_2)\big)[r, c, k]$$

at every pixel (r, c) and channel k. Alice embeds her secret into each image at an independently sampled position (r_i, c_i, k_i) from $h \times w \times 3$ possibilities.

Limitations. ProSt allocates one cover per circuit wire. Each gate reads 3 input wire images and writes 3 output wire images, so 6 images per gate, though input images are shared across gates and the distinct count is the wire count L rather than $6n$. For the 17-gate variant of the same function, $L = 61$.

ProSt also ships a pass that randomises wire names and pads the circuit with extra ancillary wires and gates. CRISP does not treat that pass as a component and claims nothing for it. Proposition 3 recovers the wiring graph from the images alone, without reading any wire name, so relabelling changes nothing an adversary can act on. We use its output only as a second benchmark size, and Remark 2 shows that the extra constant wires it introduces cost positional security rather than buying anything.

4 Threat Model and Security Goals

Homomorphic steganography occupies a different position in the security landscape from classical steganography. To avoid confusion that the unfamiliar setting can produce, we state the model precisely here so that subsequent claims can be verified against it without ambiguity.

4.1 Adversary and Knowledge

The adversary is the cloud provider Carol, who is *honest-but-curious*: she follows the protocol exactly but tries to extract information from everything she sees (Figure 1). By Kerckhoffs's principle she knows the scheme, cover format, the circuit specification including both $\pi_{\text{in}}^{(i)}$ and $\pi_{\text{out}}^{(i)}$ at every gate, and the fact that LSB steganography is in use. Withheld from her are: the per-execution secret pixel position $K = (row, col)$; the secret input triple $T = (c, x, y)$ that Alice writes at K before circuit evaluation. Active corruption, traffic injection, and collusion with Bob are out of scope.

Key delivery. K travels from Alice to Bob over a confidential out-of-band channel (Section 6.6). The permutations are public and travel with the circuit specification. Carol never receives K.

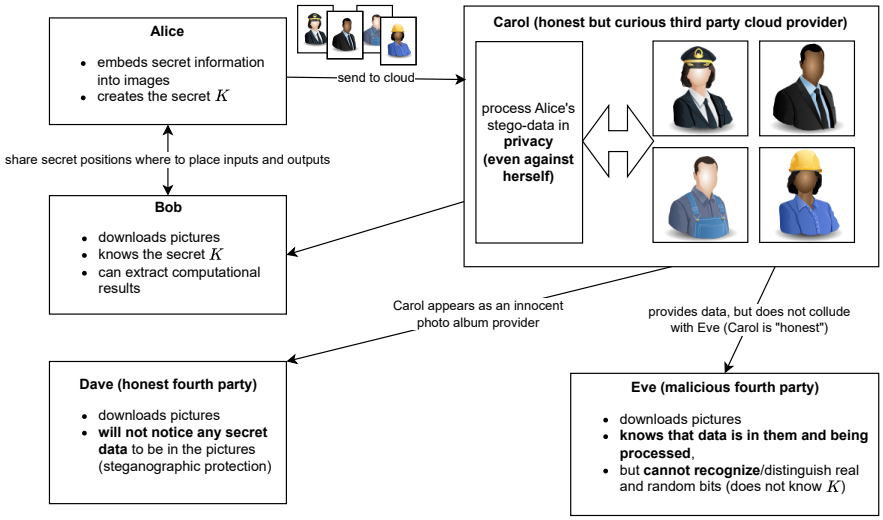

Fig. 1: System overview. **Alice** embeds secret bits at a random pixel K and uploads to **Carol**, the honest-but-curious cloud, who evaluates the circuit and forwards outputs to **Bob**. K travels Alice $\rightarrow$ Bob out of band (Section 6.6); Carol never sees it. **Dave** and **Eve** are passive third parties. The analysis targets the stronger Carol case.

4.2 Security Goals and Scope

What CRISP hides. (i) *Positional hiding*: no pixel can be distinguished as the carrier of T (Theorem 3). (ii) *Value hiding through positional uncertainty*: at the correctly guessed pixel the secret bits are indistinguishable from any other pixel's LSB triple, provided the decoy planes are i.i.d. uniform and no wire carries a value Carol already knows. Repairs R1 and R2 of Section 5.5 make both provisos hold.

What CRISP does not hide. Three things, and we are explicit about all of them. First, the existence of stego content: computing on hidden data requires the server to process it. Second, the circuit. The gate count, the wiring topology and the channel-to-role map at every gate are all available to Carol, who holds $\pi_{\text{in}}^{(i)}$ and $\pi_{\text{out}}^{(i)}$ because she needs them to evaluate and route (Section 5.4), so inverting them is free. CRISP offers no circuit privacy in any sense, scoped or otherwise, and certainly not in the simulation-based sense of garbled circuits [3]; Propositions 2 and 3 make this precise. Third, undetectability. The constant-carrier planes of repair R2 have degenerate LSB channels by design, and the re-randomised planes of R1 are uniform where natural planes are not. The model departs from Cachin-style undetectability stego [6,7] precisely here: Carol cannot be denied knowledge that the cover carries computable hidden state, so the analysis targets positional secrecy under known presence instead.

5 CRISP: Formal Scheme Definition

5.1 System Parameters

Let images have fixed dimensions $h \times w \times 3$. A Boolean circuit C is a DAG with:

- Input wires W_{in}, intermediate wires W_{mid}, output wires W_{out}
- n Fredkin gates $\mathcal{G} = \{G_1, \ldots, G_n\}$ in valid topological order w.r.t. the graph of C
- Each gate G_i has input triple (c_i, x_i, y_i) and output triple $(c_i^{\text{out}}, x_i^{\text{out}}, y_i^{\text{out}})$

5.2 Secret Key

The secret key shared between Alice and Bob is a single pixel position used uniformly across every gate of one execution:

$$K = (row, col) \quad \text{with} \quad (row, col) \in \{0, \ldots, h-1\} \times \{0, \ldots, w-1\}. \tag{4}$$

The same (row, col) is used as the embedding/extraction location at every gate G_i. This matches the implementation, which calls LocGen exactly once per execution. The permutations $\pi_{\text{in}}^{(i)}, \pi_{\text{out}}^{(i)} \in S_3$ are independently sampled per gate and included in the public circuit specification. K is communicated from Alice to Bob over a private channel before any computation. Adaptive security across multiple executions requires K to be resampled freshly per execution; the operational consequences are stated in Section 6.6.

5.3 Algorithms

The trivial samplers are stated inline: $\text{LocGen}(h, w)$ returns $(row, col) \xleftarrow{\$} \{0, \ldots, h-1\} \times \{0, \ldots, w-1\}$; $\text{PermGen}()$ returns $(\pi_{\text{in}}, \pi_{\text{out}}) \xleftarrow{\$} S_3 \times S_3$ independently. The three core operations are:

Algorithm 1: $\text{Emb}(I, c_{\text{bit}}, x_{\text{bit}}, y_{\text{bit}}, row, col, \pi_{\text{in}} \sim \text{Alice})$

$I' \leftarrow I$; $\quad bits \leftarrow (c_{\text{bit}}, x_{\text{bit}}, y_{\text{bit}})$
for $t \in \{0, 1, 2\}$ **do**
  $I'[row, col, \pi_{\text{in}}(t)] \leftarrow (I[row, col, \pi_{\text{in}}(t)] \,\&\, \texttt{0xFE}) \mid bits[t]$
return I'

Algorithm 2: $\text{Comp}(I_{\text{in}}, I_{\text{out}}, \pi_{\text{in}}, \pi_{\text{out}} \sim \text{Carol})$

$result \leftarrow I_{\text{out}}.copy()$
for each pixel (row', col') **do**
  $b_t \leftarrow I_{\text{in}}[row', col', \pi_{\text{in}}(t)] \,\&\, 1$ for $t \in \{0, 1, 2\}$
  $(c_{\text{out}}, x_{\text{out}}, y_{\text{out}}) \leftarrow F(b_0, b_1, b_2)$; $\quad out \leftarrow (c_{\text{out}}, x_{\text{out}}, y_{\text{out}})$
  for $t \in \{0, 1, 2\}$ **do**
    $result[row', col', \pi_{\text{out}}(t)] \leftarrow (result[row', col', \pi_{\text{out}}(t)] \,\&\, \texttt{0xFE}) \mid out[t]$
return $result$

Ext (Algorithm 5.3, run by Bob) is a single read: given I_{out}, the secret (row, col) and π_{out}, it returns $(I_{\text{out}}[row, col, \pi_{\text{out}}(t)] \,\&\, 1)_{t \in \{0,1,2\}}$ as $(c_{\text{out}}, x_{\text{out}}, y_{\text{out}})$.

5.4 Multi-Gate Circuit Evaluation

CRISP represents every wire as a (image, channel) pair: the image carries the wire's bit in the LSB of channel $\mathrm{ch} \in \{0,1,2\}$ at every pixel. Routing a wire to a downstream gate is a per-pixel channel-copy operation that does not depend on the secret position.

Superscript $(i) \in \{1,\dots,n\}$ indexes gates and $(t) \in \{0,1,2\}$ indexes the three logical roles of a Fredkin gate.

Wire routing via Assemble**.** For a gate G_i with public $\pi_{\text{in}}^{(i)}$ and three input carriers $\{(I^{(t)}, \mathrm{ch}^{(t)})\}_{t=0}^{2}$, Carol assembles the gate's stego image by overwriting channel $\pi_{\text{in}}^{(i)}(t)$ of a fresh cover with the LSB of channel $\mathrm{ch}^{(t)}$ of $I^{(t)}$ at every pixel; non-LSB bits and other channels remain unchanged. We denote this Assemble$(I_{\text{in}}^{(i)}, \{(I^{(t)}, \mathrm{ch}^{(t)})\}, \pi_{\text{in}}^{(i)})$ (Algorithm 3). It touches every pixel uniformly and never reads (row, col).

Algorithm 3: Assemble$(I_{\text{cover}}, \{(I^{(t)}, \mathrm{ch}^{(t)})\}_{t=0}^{2}, \pi_{\text{in}} \sim$ executed by Carol)

$I' \leftarrow I_{\text{cover}}.copy()$
for $t \in \{0,1,2\}$ **do**
  $target \leftarrow \pi_{\text{in}}(t)$
  for each pixel (row', col') **do**
    $bit \leftarrow I^{(t)}[row', col', \mathrm{ch}^{(t)}] \,\&\, 1$
    $I'[row', col', target] \leftarrow (I'[row', col', target] \,\&\, \texttt{0xFE}) \mid bit$
return I'

Full evaluation. The protocol proceeds as follows for an n-gate circuit in any (randomly selected) topological order:

1. *Key Generation.* Alice runs LocGen once, producing $K = (row, col)$ (transmitted to Bob separately; Carol never sees it). She runs PermGen per gate, producing public $(\pi_{\text{in}}^{(i)}, \pi_{\text{out}}^{(i)})$ for $i = 1,\dots,n$.
2. *Source Embedding.* Alice prepares one carrier image per primary-input wire and per *secret-valued ancillary wire*, embedding the bit at (row, col), channel 0 using Emb (Alg:1), after applying repair R1 below. Ancillary wires whose value is a public constant are handled by repair R2 and are not transmitted at all. The choice of channel 0 is arbitrary; the consuming gate's $\pi_{\text{in}}^{(i)}$ dictates the role mapping. Alice forwards the remaining source images, per-gate input covers $\{I_{\text{in}}^{(i)}\}$, and output covers $\{I_{\text{out}}^{(i)}\}$ to Carol.
3. *Computation.* Carol maintains a wire table $\mathcal{W}$ initialised from the source images. The wire table is built from the public circuit specification, so the wiring graph is known to Carol, as is the channel-to-role map at every gate (Proposition 2). For each G_i in topological order with input wires $(c_{\text{w}}, x_{\text{w}}, y_{\text{w}})$ and output wires $(c_{\text{ow}}, x_{\text{ow}}, y_{\text{ow}})$, Carol: (a) looks up the three input carriers in $\mathcal{W}$; (b) computes $stego \leftarrow$ Assemble$(I_{\text{in}}^{(i)}, \mathcal{W}[\cdot], \pi_{\text{in}}^{(i)})$; (c) computes $I_{\text{res}}^{(i)} \leftarrow$ Comp$(stego, I_{\text{out}}^{(i)}, \pi_{\text{in}}^{(i)}, \pi_{\text{out}}^{(i)})$; (d) updates $\mathcal{W}[c_{\text{ow}}] \leftarrow (I_{\text{res}}^{(i)}, \pi_{\text{out}}^{(i)}(0))$ and analogously for $x_{\text{ow}}, y_{\text{ow}}$. Carol never receives (row, col).

4. *Extraction.* For each primary output wire w, Carol forwards $\mathcal{W}[w] = (I, \mathrm{ch})$ to Bob, who applies Ext (Alg:5.3) at (row, col) on channel ch.

Fan-out. Assemble reads a wire's (image, channel) pair without consuming it, so a wire feeding k gates needs no duplication. Take w driven by G_1 and read by G_2 and G_3: after G_1 the wire table holds $\mathcal{W}[w] = (I^{(1)}_{\mathrm{res}}, \pi^{(1)}_{\mathrm{out}}(1))$, and assembling G_2 and G_3 copies that plane into two different covers without touching $I^{(1)}_{\mathrm{res}}$. Image count therefore tracks gate count, not fan-out, which is why Section 5.6 counts $2n$ plus sources rather than wires. General DAGs need only a topological order, which Assemble already assumes.

5.5 Two Repairs

The security analysis of Section 6 needs two properties that the scheme as described so far does not deliver. Both repairs are cheap and neither changes correctness.

R1: plane re-randomisation. Before writing the secret bit at K, Alice overwrites the entire LSB plane of channel 0 of each secret-carrying source image with fresh uniform bits. Without R1 the decoy bits are the cover's own LSBs, and whether those are i.i.d. uniform is a property of the cover rather than of the scheme. We measured it on a real 31-image PNG album at 128×128. Its channel-0 LSB planes carry ones 31.8% of the time against 50% under the i.i.d. uniform model, horizontally adjacent LSBs agree 69.2% of the time against 50% for an independent plane, and one image in the album has an LSB plane that is entirely constant. Cover type decides this. Camera photographs with sensor noise sit near 50.1% agreement and satisfy the assumption almost unaided, while graphics, screenshots, renders and flat or heavily quantised regions reach 99.2%, and a noiseless gradient can have an LSB plane that is identically zero, in which case one embedded bit is a single set pixel on an empty plane. R1 removes the dependence on which case a deployment falls into, for one pass over the image and nothing in communication. It also removes the residual signal that RS and sample-pair steganalysis exploit [16].

The noiseless gradient is the instructive row. Its LSB plane carries no ones at all, so the positional analysis of Section 6.4 does not apply to it in any form: Carol locates K by looking for the one pixel that differs. R1 costs the same on every row, which is why it is unconditional rather than applied only to covers that fail a test.

R2: pixel-uniform constant planes. A wire whose value is fixed and public carries that value in the LSB of its channel at *every* pixel, not only at K. Carol materialises such a plane locally from the circuit specification, so Alice transmits no carrier for it. Section 6.4 shows why this matters: a constant embedded only at K lets Carol discard every pixel whose LSB disagrees with the value she already knows, and enough constants locate K exactly. R2 removes the leak and removes $|W_{\mathrm{const}}|$ images from the transmission. The price is that a constant plane is not

Table 1: LSB-plane statistics by cover type, channel 0 at 128×128. *Set bits* is the fraction of ones, 0.5 under the i.i.d. uniform model. *Agreement* is how often horizontally adjacent LSBs match, also 0.5 when the plane is independent. The last column is the same statistic after R1.

Cover type	Set bits	Agreement	After R1
Camera photograph, sensor noise	0.5029	0.5007	0.4976
Real 31-image album, mean	0.3177	0.6917	≈ 0.5
Worst image in that album	–	1.0000	≈ 0.5
Flat or screenshot-like region	0.5000	0.9921	0.5050
Heavily quantised JPEG	0.5000	0.9921	0.4995
Noiseless gradient	0.0000	1.0000	0.5031

undetectable, since its LSB channel is degenerate. That is acceptable because the plane carries no secret and its content is public.

Implementation faithfulness. The reference implementation[1] realises the protocol exactly as described: `Carol_evaluate` takes the wire table, per-gate covers, and per-gate permutations as arguments, but *not* the secret position; only `Bob_extract` reads it. All 288 unit-test cases and the full truth-table verification of the benchmark circuit pass on this implementation.

5.6 Image Count

ProSt [1] allocates one image per circuit wire (6 per Fredkin gate: 3 inputs + 3 outputs), with input ancillaries shared across gates, giving total image count L (the wire count). CRISP allocates one fresh input cover $I_{\text{in}}^{(i)}$ and one independent output cover $I_{\text{out}}^{(i)}$ per gate (2 per gate, no sharing), plus one source image per primary-input and ancillary wire to seed the wire table.

Proposition 1 (Image Count). *A circuit of n Fredkin gates over $|W_{\text{in}}|$ primary-input wires and $|W_{\text{anc}}|$ ancillary wires requires exactly $2n + |W_{\text{in}}| + |W_{\text{anc}}|$ cover images under CRISP and L under ProSt.*

For the 14-gate benchmark, CRISP uses 33 versus ProSt's 47 (1.42×). For the 17-gate variant, 44 versus 61 (1.39×). The per-gate asymptotic reduction is $6/2 = 3\times$; source-image overhead amortises as n grows.

Two caveats. These figures count a carrier for every ancillary wire including the constant ones, so repair R2 makes them upper bounds and we report the conservative numbers. And the 14-gate figure is the Fredkin synthesis published with ProSt, not a lower bound; a tighter synthesis of the same function needs fewer gates and narrows the gap, since ProSt's count scales with wires and CRISP's with gates. Read the result as a crossover, CRISP wins once $2n + |W_{\text{in}}| + |W_{\text{anc}}| < L$, rather than as a fixed speedup measured on one small benchmark.

[1] https://github.com/shahzadssg/CRISP.git

5.7 Correctness

Theorem 1 (Correctness). *For any circuit C, any secret input bits, and any key K, the output recovered by* Ext *(Alg:5.3) equals the correct evaluation of C.*

Proof. Induct over gates in topological order on the invariant that after G_i executes, the wire table assigns each output wire w a carrier (I, ch) with $I[row, col, \mathrm{ch}] \,\&\, 1$ equal to the correct logical value of w. It holds at the sources by construction, since Alice writes each secret-valued wire into channel 0 at (row, col) and R2 makes each public-constant wire a pixel-uniform plane. For the step, Assemble places input role t in channel $\pi_{\mathrm{in}}^{(i)}(t)$, so at (row, col) the assembled cover carries (c, x, y) in the roles G_i expects; Comp reads those channels, applies F, and writes output role t into channel $\pi_{\mathrm{out}}^{(i)}(t)$, which is where Carol points the three output wires. After the last gate Bob reads $\mathcal{W}[w]$ at (row, col). Carol's processing is pixel-uniform, so that bit is computed exactly as every other pixel's; only Bob's secret position selects the one that means anything. The step-by-step induction is in Appendix A. □

6 Security Analysis

6.1 Security Model

We analyse CRISP against an honest-but-curious adversary Carol who receives all stego images, all source images, all per-gate input and output covers, and the full circuit description including $\pi_{\mathrm{in}}^{(i)}$ and $\pi_{\mathrm{out}}^{(i)}$ for every gate (both are required for correct evaluation). Carol does not receive the per-execution secret position (row, col), which is shared between Alice and Bob over a private channel before computation begins.

Definition 2 (CRISP Security Game $\mathrm{Game}_{\mathrm{CRISP}}$**).** *The challenger fixes h, w, samples a fresh $K = (row, col)$, chooses a secret input T, and runs the protocol. Carol receives the full transcript $\mathcal{T}$ (Definition 3) and the circuit description including every $\pi_{\mathrm{in}}^{(i)}, \pi_{\mathrm{out}}^{(i)}$, then outputs a position guess (row^*, col^*) and a value guess T^*. She wins the joint game when both are correct, and the value-only game when $T^* = T$ after committing to a position before seeing $\mathcal{T}$. The single-image case $\mathcal{T} = (I')$ is the warm-up of Theorem 2; the general case is Theorem 3. The corresponding advantages are*

$$\begin{aligned} \mathrm{Adv}_{\mathcal{A}}^{\mathsf{val}} &= \left|\Pr[T^* = T \mid M = m,\ (row^*, col^*)\ \textit{fixed}] - \Pr[T^* = T]\right|, \\ \mathrm{Adv}_{\mathcal{A}}^{\mathsf{joint}} &= \Pr\left[\mathsf{Win}^{\mathsf{joint}}\right], \end{aligned}$$

where M denotes the observed image LSBs and m is an arbitrary realisation. Theorem 2 bounds $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{val}}$ for one image; Theorem 3 gives an exact expression for $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{joint}}$ over the whole transcript.

6.2 Circuit Privacy: A Negative Result

Resampling π_{out} per gate looks like it should buy circuit privacy. It does not, and we state that plainly here, because the intuition behind it is a natural one and the same trick will occur to anyone building computation on top of permuted channel roles.

By *per-gate resampling* we mean that each $\pi_{\text{out}}^{(i)}$ is drawn uniformly from S_3, independently of $\pi_{\text{in}}^{(i)}$ and of the permutations at every other gate.

Proposition 2 (No circuit privacy against a specification holder). *Let Carol hold the public circuit specification, including $\pi_{\text{in}}^{(i)}$ and $\pi_{\text{out}}^{(i)}$ for every gate G_i. Then Carol learns the channel-to-role map at the input and at the output of every gate, and hence the control wire of every gate. This holds whether π_{out} is fixed once, resampled per gate, or resampled per pixel.*

Proof. Carol needs $\pi_{\text{in}}^{(i)}$ to read the gate's logical inputs and $\pi_{\text{out}}^{(i)}$ to look up the gate's outputs in the wire table when routing to downstream gates (Section 5.4). Both are therefore in her possession by construction, not by attack. Applying $(\pi_{\text{in}}^{(i)})^{-1}$ and $(\pi_{\text{out}}^{(i)})^{-1}$ restores the canonical role layout at G_i in constant time. The wiring graph is public, so the restored layouts compose across the whole circuit. Freshness of $\pi_{\text{out}}^{(i)}$ is irrelevant: a permutation the adversary is handed is not a secret at any resampling rate. □

Withholding the permutations is not an option. Carol cannot evaluate the circuit without π_{in} and cannot route wires without π_{out}. The next result shows that withholding would not help anyway.

Proposition 3 (Role and wiring recovery without the specification). *Let $\mathcal{E}$ hold the image set but not the circuit specification. For each gate, $\mathcal{E}$ identifies the input control channel and the output control channel using nine channel-pair comparisons of $O(hw)$ bit operations each, with error probability at most $(3/4)^{hw}$. Repeating the same statistic across image pairs recovers the wiring graph.*

Proof. For an input image I_{in}, a result image I_{res}, and channels $a, b \in \{0, 1, 2\}$, let

$$S_{a,b} \;=\; \sum_{row', col'} \Big(I_{\text{in}}[row', col', a] \;\&\; 1\Big) \oplus \Big(I_{\text{res}}[row', col', b] \;\&\; 1\Big).$$

The Fredkin gate satisfies $c_{\text{out}} = c$ for all inputs, so the pair (a, b) carrying the control bit at input and at output gives $S_{a,b} = 0$ at every pixel. For the other eight pairs, enumerate the input triples under uniform (c, x, y) and write (x', y') for the data outputs, so $(x', y') = (x, y)$ when $c = 0$ and (y, x) when $c = 1$.

Input control against a data output. $c \oplus x'$ equals x when $c = 0$ and $1 \oplus y$ when $c = 1$, each of which is 1 with probability $1/2$, so the pair disagrees with probability $1/2$. The same holds for $c \oplus y'$.

A data input against a data output. $x \oplus x'$ equals 0 when $c = 0$ and $x \oplus y$ when $c = 1$, so it is 1 with probability $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Symmetrically $x \oplus y'$ equals

$x \oplus y$ when $c = 0$ and 0 when $c = 1$, again 1/4. The remaining data pairs follow by exchanging x and y.

A data input against the control output. $x \oplus c$ is 1 with probability 1/2.

Every wrong pair therefore disagrees at each pixel with probability at least 1/4, so it survives all hw pixels with probability at most $(3/4)^{hw}$, below 2^{-1000} already at 64×64. Nine pairs, one pass each. The same statistic recovers the wiring graph: Assemble copies LSB planes verbatim, so the result image of G_i and the assembled input image of G_j agree exactly on a channel pair whenever a wire runs from G_i to G_j, and $S_{a,b} = 0$ identifies it. □

The two propositions together say that CRISP has no circuit privacy against any party who sees the images. Table 5 records this. The single-image embedding and the image-count reduction stand on their own; the permutations remain in the scheme because they cost nothing and because π_{in} is needed to place three roles into three channels, not because they hide anything.

Remark 1 (What would be needed, and what it costs). It is fair to ask whether simulation-based circuit privacy is reachable inside this framework. It is not, as long as the server routes wires by reading a specification. Two routes exist and both cost more than the single-image embedding saves. Evaluating a universal circuit and carrying the program as data pushes the image count past the wire-per-image baseline, since Valiant's construction has size $\Theta(|C| \log |C|)$ [21], and makes the program encoding a second thing to hide. Oblivious routing turns each wire lookup into an ORAM access and reintroduces the interaction that asynchronous outsourcing avoids. Either way Section 6.4 carries over, since it never depended on the permutations.

6.3 Information Hiding: Bayesian Advantage Bound

The main security result establishes that the adversary's advantage in inferring the embedded secret decays as $\Theta(1/n)$ with $n = h \times w$. The proof is self-contained and rests on a direct Bayesian calculation.

Theorem 2 (Information Hiding for CRISP). *Let $p_0 \in (0, 1)$, and let $T = (c, x, y) \sim \mathrm{Ber}(p_0)^3$ be a secret bit-triple. CRISP embeds T at a uniformly random pixel (row^*, col^*) chosen from $n = h \times w$ positions; all remaining pixel LSB triples are drawn i.i.d. from $\mathrm{Ber}(p_0)^3$, independently of T. Then*

$$\mathrm{Adv}_{\mathcal{A}} = \frac{1 - p_t}{n} \;\in\; \Theta\left(\frac{1}{n}\right), \quad n = h \times w,$$

where $p_t = \Pr(T = t) = p_0^{w(t)}(1 - p_0)^{3-w(t)}$ for triple t with Hamming weight $w(t)$, and $(1 - p_t) \in (0, 1)$ is a positive constant.

Proof. **Setup.** Represent the image LSBs as a vector $\mathbf{M} = (M_0, M_1, \ldots, M_{n-1})$ of pixel triples $M_j \in \{0, 1\}^3$. Since Carol knows π_{in}, she recovers the logical ordering of each pixel's three bits directly. The secret triple T sits at a position j^*

drawn uniformly and independently from $\{0, \ldots, n-1\}$, so $\Pr(j^* = j) = 1/n$ for all j. Every position $j \neq j^*$ holds an i.i.d. decoy triple from $\mathrm{Ber}(p_0)^3$, independent of T. Since T follows the same law, $\mathbf{M}$ is *exchangeable*: every permutation of its coordinates has the same joint distribution.

Adversary's optimal strategy. By exchangeability no position is distinguishable a priori, so without loss of generality Carol inspects position 0. Her optimal value guess is $T^* = M_0$, that is, guess that position 0 holds the secret and that its value equals the observed triple.

Marginal of M_0. For any triple $t \in \{0,1\}^3$,

$$\Pr(M_0 = t) = \Pr(j^* = 0)\Pr(M_0 = t \mid j^* = 0) + \Pr(j^* \neq 0)\Pr(M_0 = t \mid j^* \neq 0).$$

When $j^* = 0$ we have $M_0 = T$, so the first conditional is p_t. When $j^* \neq 0$, M_0 is an independent decoy, so the second conditional is also p_t. Hence $\Pr(M_0 = t) = \frac{1}{n}p_t + \frac{n-1}{n}p_t = p_t$: a decoy and the secret are marginally identical, which is the whole point of the construction.

Joint probability. Conditioning the same way,

$$\begin{aligned}\Pr(T = t,\, M_0 = t) = \Pr(j^* = 0)\Pr(T = t,\, M_0 = t \mid j^* = 0) \\ + \Pr(j^* \neq 0)\Pr(T = t,\, M_0 = t \mid j^* \neq 0).\end{aligned}$$

If $j^* = 0$ then $M_0 = T$, so the event reduces to $T = t$ and the conditional is p_t. If $j^* \neq 0$ then T and M_0 are independent and the conditional is p_t^2. Therefore

$$\Pr(T = t,\, M_0 = t) = \frac{1}{n}p_t + \frac{n-1}{n}p_t^2 = p_t\left(\frac{1}{n} + \frac{n-1}{n}p_t\right).$$

Posterior and advantage. Bayes' theorem gives

$$\Pr(T = t \mid M_0 = t) = \frac{p_t\left(\frac{1}{n} + \frac{n-1}{n}p_t\right)}{p_t} = \frac{1}{n} + \frac{n-1}{n}p_t = p_t + \frac{1-p_t}{n},$$

and the prior probability of a correct value guess is p_t, so $\mathrm{Adv}_{\mathcal{A}} = (1-p_t)/n$. For any $p_0 \in (0,1)$ the factor $(1-p_t)$ is a positive constant independent of n, so $\mathrm{Adv}_{\mathcal{A}} \in \Theta(1/n)$. □

This is a one-image, one-triple statement. A circuit transcript holds many images and many wires, some of whose values Carol already knows, so the extension to a full circuit is not a formality. Section 6.4 redoes the analysis there.

Comparison with ProSt. ProSt embeds one scalar bit at one of $n_P = h{\times}w{\times}3$ pixel-channel positions ($\mathrm{Adv} \in \Theta(1/n_P)$); CRISP embeds a triple at one of $n_C = h{\times}w$ pixel positions ($\mathrm{Adv} \in \Theta(1/n_C)$). For 128×128, $n_P \approx 49\,000$ and $n_C \approx 16\,000$; both bounds are statistically small but *not* cryptographically negligible (the decay is polynomial in n, not super-polynomial). CRISP trades ProSt's tighter individual bound for a 1.39–1.42× image-count reduction. It buys no circuit privacy in exchange, for the reasons given in Section 6.2. Note also that ProSt embeds its ancillary constants the same way CRISP originally did, so Corollary 1 applies to ProSt as well; repair R2 is not a CRISP-specific patch.

6.4 Joint Recovery over the Full Transcript

Theorem 2 treats one image and one free triple. Carol sees far more: every source carrier, every assembled gate input, and every gate result. We restate the game over that whole object, because a bound proved on one image does not extend to a circuit by assertion, and here it does not extend at all unless the scheme is repaired.

Definition 3 (Transcript). *The* transcript $\mathcal{T}$ *is the ordered tuple of every image Carol receives or produces in one execution: the source carriers, the per-gate assembled inputs* $\{stego^{(i)}\}$*, and the per-gate results* $\{I_{\mathrm{res}}^{(i)}\}$*. Write* $\mathcal{T}[j]$ *for the restriction of* $\mathcal{T}$ *to pixel* j*, that is, the LSB triple of every image in* $\mathcal{T}$ *at that pixel.*

Lemma 3 (Pixel-wise independence). *Under repairs R1 and R2 (Section 5.5) the slices* $\mathcal{T}[0], \ldots, \mathcal{T}[n-1]$ *are mutually independent. For* $j \neq K$*,* $\mathcal{T}[j]$ *is the transcript of a complete evaluation of the same circuit on an assignment drawn uniformly from* $\{0,1\}^m$*, where* m *is the number of secret-valued source wires.* $\mathcal{T}[K]$ *is the evaluation on the true assignment* T*.*

Proof. Emb, Assemble and Comp act pixel by pixel and never read or write across pixels, so every image in $\mathcal{T}$ is a deterministic pixel-wise function of the source planes. R1 makes the channel-0 plane of each secret carrier i.i.d. uniform away from K, so assignments at distinct pixels are independent and uniform on $\{0,1\}^m$. R2 gives each public-constant wire the same value at every pixel, so it contributes no cross-pixel variation and no dependence between slices. □

Theorem 3 (Joint position and value recovery). *Let* q *be Carol's prior on the secret assignment* $T \in \{0,1\}^m$ *and suppose* q *is uniform on a support* $S \subseteq \{0,1\}^m$*, so that* $\lambda := m - H_2(q) = m - \log_2 |S|$ *with* H_2 *the Rényi collision entropy. Under Lemma 3, Carol's optimal success probability in the joint game of Definition 2, played over the whole transcript, is*

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{joint}} \;=\; \mathbb{E}\left[\frac{1}{1+B}\right], \qquad B \sim \mathrm{Bin}\big(n-1,\; 2^{-\lambda}\big)\,, \qquad n = h{\times}w,$$

and therefore

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{joint}} \;\geq\; \frac{2^{\lambda}}{2^{\lambda}+n-1}.$$

When $\lambda = 0$ *both expressions equal* $1/n$*, recovering the earlier claim.*

Proof. The slice $\mathcal{T}[j]$ contains the LSBs of the secret-valued source carriers at pixel j, which is exactly the assignment $v_j \in \{0,1\}^m$; every other image in the slice is a deterministic function of v_j. By Lemma 3, $v_K \sim q$ and $v_j \sim \mathrm{Unif}(\{0,1\}^m)$ independently for $j \neq K$, with K uniform on the n pixels and independent of everything else. Carol's posterior on the secret pixel is

$$\Pr[K = j \mid \mathcal{T}] \;\propto\; q(v_j) \prod_{i \neq j} 2^{-m} \;\propto\; q(v_j),$$

which is uniform on the candidate set $\mathcal{C} = \{\, j : v_j \in S \,\}$ and zero outside it. All candidates are equally good, and committing to a pixel j fixes the value guess to v_j, which is correct exactly when $j = K$. So Carol wins with probability $1/|\mathcal{C}|$, and no other strategy does better. Now $K \in \mathcal{C}$ always, and each of the remaining $n-1$ pixels joins independently with probability $|S|2^{-m} = 2^{-\lambda}$, giving $|\mathcal{C}| = 1 + B$ with $B \sim \mathrm{Bin}(n-1, 2^{-\lambda})$. Jensen's inequality on the convex map $x \mapsto 1/(1+x)$ gives $\mathbb{E}[1/(1+B)] \geq 1/(1+\mathbb{E}[B]) = 2^{\lambda}/(2^{\lambda}+n-1)$. □

The parameter λ has a plain reading: it is the number of bits of the secret input that Carol already knows. Every such bit halves the candidate set. Uniform inputs give $\lambda = 0$ and the exact $1/n$ bound; a fully known input gives $\lambda = m$ and no positional security at all once $2^m \geq n$.

Corollary 1 (Cost of public-constant wires). *Suppose R2 is not applied and κ ancillary wires carry publicly known constants embedded only at K. Then $\lambda \geq \kappa$, and $\mathrm{Adv}^{\mathsf{joint}}_{\mathcal{A}} \geq 2^{\kappa}/(2^{\kappa}+n-1)$.*

Proof. A constant wire is a source wire whose value Carol knows, so q is a point mass on that coordinate and it contributes one full bit to $m - H_2(q)$. The constraint is testable pixel by pixel: any j whose LSB in that carrier disagrees with the known value leaves $\mathcal{C}$. R1 makes the κ constraints independent. □

Remark 2 (Scale of the leak). What sets κ is the number of distinct constant carriers in the circuit, and nothing else. The 14-gate benchmark carries two, so $\kappa = 2$ and the candidate set shrinks by a factor of four. The 17-gate variant carries seven, so $\kappa = 7$, and at 128×128 the bound is $128/(128+16383) \approx 1/129$ rather than the $1/16384$ a single-image analysis suggests. Running the attack against the reference implementation gives a mean candidate set of 128.6 over 60 executions, which matches.

Wires whose values follow from the constants by propagation add nothing further, which is worth stating because it is easy to assume otherwise. Any pixel that already survives the filters on a gate's constant inputs reproduces that gate's outputs automatically, so the derived constraints are implied rather than independent. In the 17-gate variant, 16 wires have publicly derivable values and the measured filtering is still 2^7. Padding a circuit with extra constants therefore costs security in direct proportion to how many distinct carriers it adds. That is the concrete reason R2 is not optional.

6.5 Hamming Weight Preservation and Security Under Computation

Lemma 4 (Distribution Preservation). *The pixel-level LSB Hamming-weight distribution is unchanged by* Comp*: for every pixel (row', col'), $\sum_k \mathrm{lsb}(I_{\mathrm{out}}[row', col', k]) = \sum_k \mathrm{lsb}(I_{\mathrm{in}}[row', col', k])$.*

Proof. Lemma 2 guarantees F preserves Hamming weight; Comp applies F pixel-wise and the permutations $\pi_{\mathrm{in}}, \pi_{\mathrm{out}}$ leave the per-pixel weight unchanged. □

Corollary 2 (Adding gates does not degrade the bound). *Theorem 3 already quantifies over the whole circuit, so no separate gate-by-gate composition step is required. Extra gates add images to $\mathcal{T}$, but by Lemma 3 each added image is a pixel-wise function of assignments already accounted for, so the candidate set $\mathcal{C}$ is unchanged. The bound depends on n and λ only, not on the gate count.*

6.6 Naïve Extraction, Steganalysis and Repeated Use

Concatenation. Under R1 the channel-0 plane is i.i.d. uniform, so concatenating all $3hw$ LSBs gives a string in which the secret triple is distributed exactly like its $3n-3$ decoys and no locator information survives. Without R1 the decoys are natural-cover LSBs, neither uniform nor independent across neighbours, and the uniform embedded triple stands out; that is the standard LSB-replacement signal [16]. Theorem 3 presumes R1.

Localised steganalysis. Comp touches every pixel, so there is no localised modification footprint to exploit, and Lemma 4 leaves the per-pixel Hamming-weight distribution unchanged. Positional secrecy is independent of detectability and survives detection of stego presence.

Repeated use. Across executions, K must be resampled freshly and covers must not be reused, since two runs on one cover differ only where the LSB planes disagree. A 17-gate circuit at 128×128 needs about $\log_2(hw) + n \cdot 2\log_2 6 \approx 102$ bits of fresh randomness per execution from a PRG seeded by a long-term Alice–Bob secret. Alice sends K over a confidential out-of-band channel, and deriving K and the permutation stream from one seed via HKDF keeps that to a single long-term secret. Reusing K is a catastrophic failure and should be rejected at the API boundary.

7 Implementation

CRISP is implemented in Python 3.10 using NumPy 1.24 (`https://numpy.org`) and Pillow 9.4 (`https://python-pillow.org`). The complete source code is publicly available at `https://github.com/shahzadssg/CRISP.git`. It passes all 288 unit-test combinations ($8 \times 6 \times 6$ over input bits and permutations) and the full 8-row truth-table check of the benchmark $f(A, B, C) = (A \wedge C) \vee (\neg A \wedge B) \vee (\neg B \wedge \neg C)$. `Carol_evaluate` does not take the secret position as an argument; only `Bob_extract` reads it.

8 Experimental Results

Experiments were run on an Intel Core i5-1135G7 @ 2.40 GHz, 16 GB RAM, Python 3.10, NumPy 1.24, Pillow 9.4, with timings averaged over 100 runs and the 17-gate variant of the benchmark circuit for ProSt comparison.

Correctness and Hamming weight. All 288 unit-test combinations pass at every image size; Hamming weight is preserved across all 1,700 gate evaluations

(100 trials × 17 gates), confirming Theorem 1 and Lemma 4. Hamming-weight preservation says nothing about Theorem 3, which we test separately below.

Single-gate timing. Table 2 compares a naive per-pixel Python loop against the vectorised NumPy implementation. Runtime scales linearly in pixel count, as a per-pixel operation should.

Table 2: Single-gate Comp time, averaged over 100 runs. Speedup = Loop/Vec.

Image	Pixels	Loop [s]	Vec. [s]	Speedup
64×64×3	12,288	0.089	0.0008	111×
128×128×3	49,152	0.341	0.0012	284×
256×256×3	196,608	1.387	0.0041	338×
512×512×3	786,432	5.521	0.0159	347×

Multi-gate timing and image count. Table 3 reports CRISP vs. ProSt-estimated timing and image counts. CRISP timings are measured on the reference implementation and include the protocol-faithful Assemble step. The ProSt column is *not* a measurement. It is a cost model that charges 3×Comp per gate, on the grounds that ProSt touches three input and three output wire images where CRISP touches one of each, and it omits ProSt's per-wire bookkeeping. We label it *ProSt-model* rather than presenting it as a baseline. Both systems are pixel-parallel and memory-bound, so the ratio mainly reflects the image-count ratio and should be read as such. A like-for-like timing against the released ProSt implementation, on the same covers and hardware, is the right experiment and we did not run it.

Table 3: Multi-gate evaluation at 128×128. CRISP times are measured; the ProSt column is the 3×Comp cost model described in the text, not a measurement.

Circuit	Gates	Wires	CRISP img	ProSt img	CRISP [ms]	ProSt-model [ms]	Ratio
Benchmark, 14 gates	14	47	33	47	6.4	13.7	2.16×
Variant, 17 gates	17	61	44	61	7.7	16.6	2.16×
Linear, 50 gates	50	155	105	155	22.5	49.0	2.18×
Linear, 100 gates	100	305	205	305	45.1	98.0	2.18×

Positional security, $\lambda = 0$. With repairs R1 and R2 in place, we simulate Carol's optimal guessing attack over 10 000 trials per image size. Joint-win rates match $1/(h{\times}w)$ within statistical noise (standard error $\approx 1.2 \times 10^{-3}$ at the smallest size): for 8×8, 16×16, 32×32, 64×64 and 128×128 the empirical rates are 1.548×10^{-2}, 3.891×10^{-3}, 9.780×10^{-4}, 2.450×10^{-4} and 6.110×10^{-5}, against $1/n$ of $1.563{\times}10^{-2}$, $3.906{\times}10^{-3}$, $9.766{\times}10^{-4}$, $2.441{\times}10^{-4}$, $6.104{\times}10^{-5}$.

Positional security, $\lambda > 0$. Table 4 measures what constant wires actually cost. We run the filtering attack of Corollary 1 over 4×10^6 trials per cell: Carol keeps every pixel whose LSB agrees with all κ known constants, then guesses uniformly among survivors. The measured rates track $\mathbb{E}[1/(1{+}B)]$ to three digits and the Jensen bound $2^\kappa/(2^\kappa + n - 1)$ is tight while $2^\kappa \ll n$. At 128×128, seven

constant wires cost 128× and fourteen leave Carol winning outright most of the time. R2 sets $\kappa = 0$ and returns the top row.

Table 4: Measured joint-recovery rate under the constant-wire filtering attack, 4×10^6 trials per cell. The last column is the measured rate divided by $1/n$, that is, how much worse the truth is than the single-image claim.

$n = h \times w$	κ	measured	$\mathbb{E}[1/(1+B)]$	$2^\kappa/(2^\kappa+n-1)$	vs. $1/n$
4096	0	2.40×10^{-4}	2.44×10^{-4}	2.44×10^{-4}	1.0×
4096	1	4.86×10^{-4}	4.88×10^{-4}	4.88×10^{-4}	2.0×
4096	3	1.97×10^{-3}	1.95×10^{-3}	1.95×10^{-3}	8.1×
4096	5	7.82×10^{-3}	7.81×10^{-3}	7.75×10^{-3}	32×
4096	7	3.12×10^{-2}	3.13×10^{-2}	3.03×10^{-2}	128×
4096	10	2.45×10^{-1}	2.45×10^{-1}	2.00×10^{-1}	1004×
4096	14	8.85×10^{-1}	8.85×10^{-1}	8.00×10^{-1}	3625×
16384	0	6.28×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.0×
16384	1	1.20×10^{-4}	1.22×10^{-4}	1.22×10^{-4}	2.0×
16384	3	4.67×10^{-4}	4.88×10^{-4}	4.88×10^{-4}	7.6×
16384	5	1.97×10^{-3}	1.95×10^{-3}	1.95×10^{-3}	32×
16384	7	7.81×10^{-3}	7.81×10^{-3}	7.75×10^{-3}	128×
16384	10	6.24×10^{-2}	6.25×10^{-2}	5.88×10^{-2}	1022×
16384	14	6.32×10^{-1}	6.32×10^{-1}	5.00×10^{-1}	10352×

Empirical steganalysis. See Appendix B.

Communication cost. This is the number that decides whether the approach is usable, so we state it in the text rather than leaving it inside a table. The 17-gate variant at 128×128 moves about 2.06 MB under CRISP against about 2.86 MB under ProSt. That is 2 MB to evaluate a three-input Boolean function. TFHE moves tens of kilobytes for the same job, three orders of magnitude less, and a TEE moves about a kilobyte. The image-count reduction is real but it improves a constant on a cost that is already far larger than the alternatives. Anyone comparing on bandwidth alone should pick FHE. The case for homomorphic steganography rests on the properties in Table 5, mainly the absence of server-held keys and of hardware roots of trust, and on the fact that the covers are ordinary images moving over ordinary channels.

Comparison with FHE and TEEs. Table 5 compares CRISP against ProSt, FHE and TEEs along the axes that dominate deployment decisions. TEEs (Intel SGX, AMD SEV, ARM TrustZone) provide hardware-attested execution and carry a documented side-channel history [5, 19]. Neither CRISP nor ProSt offers circuit privacy, so on that axis FHE with circuit-private bootstrapping is the only entry in the table that does.

Table 5: Comparison of CRISP with related work. Communication figures are for a 17-gate circuit on 128×128 covers (CRISP/ProSt) and a single 1-bit input under TFHE LWE parameters (FHE). The CRISP positional row assumes repairs R1 and R2, which give $\lambda = 0$; without them λ grows by one bit per public-constant wire.

Property	ProSt	CRISP	FHE (TFHE)	TEE (SGX)
Hides secret location	Yes	Yes	N/A (cipher hides value)	No
Circuit privacy	No	No (Prop. 2)	Yes (with circuit-private bootstrapping)	No
Server-side keys held	No	No	Yes (eval/bootstrap)	No
Positional security	$1/(h{\times}w{\times}3)$	$2^{\lambda}/(2^{\lambda}{+}hw{-}1)$	cryptographic (LWE)	hardware attestation
Images per gate	6	2	N/A	N/A
Comm. for 17-gate, 3-bit input	≈ 2.86 MB	≈ 2.06 MB	∼10–30 KB	~ 1 KB
Hardware dependency	No	No	No	Yes (specific CPU)
Interaction required	No	No	No	No (post-attestation)
Side-channel exposure	None inherent	None inherent	None inherent	Documented [5, 19]
Setup complexity	Low	Low	High (key gen, bootstrap)	High (attestation)
Adversary model assumed	Honest-but-curious	Honest-but-curious	Semantically secure	Hardware root of trust
Setup complexity	Low	Low	High (key gen, bootstrap)	High (attestation)

9 Conclusion

CRISP embeds all three inputs of a Fredkin gate into the three RGB channels of one cover and all three outputs into another, so a gate costs two images instead of ProSt's six. For the 14-gate benchmark that is 33 images against 47, and for the 17-gate variant 44 against 61. Two limits of the design matter more than that reduction. Permutations that the server must hold in order to compute give no circuit privacy, at any resampling rate, and an adversary holding only the images recovers the same information with nine comparisons per gate (Section 6.2). A $1/n$ positional bound proved on a single image does not lift to a circuit for free: wires whose values the server already knows shrink the candidate pixel set by half apiece, and a circuit padded with extra constants carries more of them. The bound that does hold over the full transcript is $2^{\lambda}/(2^{\lambda} + n - 1)$, with λ the shortfall between the secret input length and the collision entropy of the server's prior (Theorem 3), and two cheap repairs return λ to zero.

Limitations. The advantage decays polynomially in n, not super-polynomially, so the bound is statistically small and not cryptographically negligible. Scaling n costs bandwidth linearly, and 2 MB for a three-input function is already three

orders of magnitude above TFHE. The analysis assumes an honest-but-curious server; an active one who edits covers between gates is out of scope and would need cover commitments or Hamming-weight MACs. Repair R1 turns the i.i.d. decoy assumption into a property of the scheme, at the cost of an LSB plane that is uniform rather than natural. Detectability was tested only with chi-square on synthetic covers.

Future work. A like-for-like timing against the released ProSt implementation would replace the cost model in Table 3. Generalising to Toffoli gates and to audio or video covers is notational. The open direction is whether any homomorphic-steganography construction can hide the computation as well as its position, given that the server must route wires by reading something; Remark 1 gives the two candidate routes and their cost.

References

1. Ahmad, S., Rass, S.: Prost: Provably secure homomorphic steganography. IEEE Access **14**, 14393–14409 (2026). https://doi.org/10.1109/ACCESS.2026.3656995
2. Baluja, S.: Hiding images in plain sight: Deep steganography. In: Neural Information Processing Systems (2017)
3. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. p. 784–796. CCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2382196.2382279, https://doi.org/10.1145/2382196.2382279
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. p. 309–325. ITCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2090236.2090262, https://doi.org/10.1145/2090236.2090262
5. Bulck, J.V., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: Foreshadow: Extracting the keys to the intel SGX kingdom with transient Out-of-Order execution. In: 27th USENIX Security Symposium (USENIX Security 18). p. 991–1008. USENIX Association, Baltimore, MD (Aug 2018), https://www.usenix.org/conference/usenixsecurity18/presentation/bulck
6. Cachin, C.: Entropy Measures and Unconditional Security in Cryptography. Ph.D. thesis, ETH Zurich (1997), reprint as vol. 1 of ETH Series in Information Security and Cryptography, ISBN 3-89649-185-7, Hartung-Gorre Verlag, Konstanz, 1997
7. Cachin, C.: An information-theoretic model for steganography (2000), https://eprint.iacr.org/2000/028
8. Chan, C.K., Cheng, L.M.: Hiding data in images by simple lsb substitution. Pattern Recognit. **37**, 469–474 (2004)
9. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International Conference on the Theory and Application of Cryptology and Information Security (2017)
10. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. Cryptology ePrint Archive, Paper 2016/870 (2016), https://eprint.iacr.org/2016/870

11. Costan, V., Devadas, S.: Intel SGX Explained (2016), https://eprint.iacr.org/2016/086, publication info: Preprint.
12. Damgård, I., Pastro, V., Smart, N.P., Zakariás, S.: Multiparty computation from somewhat homomorphic encryption. In: IACR Cryptology ePrint Archive (2012)
13. Ding, J., Chen, K., Wang, Y., Zhao, N., Zhang, W., Yu, N.: Discop: Provably secure steganography in practice based on "distribution copies". In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 2238–2255 (2023). https://doi.org/10.1109/SP46215.2023.10179287
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144 (2012), https://eprint.iacr.org/2012/144
15. Fredkin, E., Toffoli, T.: Conservative Logic. In: Adamatzky, A. (ed.) Collision-Based Computing, pp. 47–81. Springer, London (2002). https://doi.org/10.1007/978-1-4471-0129-1_3, https://doi.org/10.1007/978-1-4471-0129-1_3
16. Fridrich, J.: Steganography in Digital Media: Principles, Algorithms, and Applications. Cambridge University Press, Cambridge (2009). https://doi.org/10.1017/CBO9781139192903, https://www.cambridge.org/core/books/steganography-in-digital-media/2D5989F2462923428BE01F2042A532AB
17. Gentry, C.: Fully homomorphic encryption using ideal lattices (2009). https://doi.org/10.1145/1536414.1536440, https://www.semanticscholar.org/paper/6e6e67042c647b8dd6818f893cbe1ffd474636a3
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. p. 218–229. STOC '87, Association for Computing Machinery, New York, NY, USA (1987). https://doi.org/10.1145/28395.28420, https://doi.org/10.1145/28395.28420
19. Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., Hamburg, M.: Meltdown: Reading kernel memory from user space. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 973–990. USENIX Association, Baltimore, MD (Aug 2018), https://www.usenix.org/conference/usenixsecurity18/presentation/lipp
20. Luo, W., Huang, F., Huang, J.: Edge adaptive image steganography based on lsb matching revisited. IEEE Transactions on Information Forensics and Security **5**(2), 201–214 (2010). https://doi.org/10.1109/TIFS.2010.2041812
21. Valiant, L.G.: Universal circuits (preliminary report). In: Proceedings of the Eighth Annual ACM Symposium on Theory of Computing (STOC '76). pp. 196–203. Association for Computing Machinery, New York, NY, USA (1976). https://doi.org/10.1145/800113.803649
22. Wu, D.C., Tsai, W.H.: A steganographic method for images by pixel-value differencing. Pattern Recognit. Lett. **24**, 1613–1626 (2003)
23. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167 (1986). https://doi.org/10.1109/SFCS.1986.25
24. Zhang, X., Chen, K., Zhao, N., Zhang, W., Yu, N.: Provably Secure Public-Key Steganography Based on Admissible Encoding. IEEE Transactions on Information Forensics and Security **20**, 3161–3175 (2025). https://doi.org/10.1109/TIFS.2025.3550076, http://arxiv.org/abs/2504.19454, arXiv:2504.19454 [cs]
25. Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L.: Hidden: Hiding data with deep networks. In: European Conference on Computer Vision (2018)

Appendix

A. Theorem 1 (Correctness), full proof

Proof. We prove correctness inductively over gates in topological order. The invariant is: after gate G_i executes, the wire table $\mathcal{W}$ assigns each output wire w of G_i a carrier (I, ch) such that $I[row, col, \mathrm{ch}] \,\&\, 1$ equals the correct logical bit value of w.

Base case (source wires). For each secret-valued source wire w, Alice prepares one image via Emb (Alg. 1), writing the wire's bit into the LSB of channel 0 at (row, col) after the plane re-randomisation of repair R1. For each public-constant wire, repair R2 supplies a pixel-uniform plane carrying the constant at every pixel, so in particular at (row, col). The wire table is initialised with $\mathcal{W}[w] = (I_w, 0)$ and the invariant holds at w by construction.

Inductive step (gate G_i). Assume the invariant holds for every wire already in $\mathcal{W}$ when G_i is processed. Carol calls Assemble (Alg. 3) with the carriers of G_i's three input wires and $\pi_{\mathrm{in}}^{(i)}$. The resulting image satisfies $stego[row, col, \pi_{\mathrm{in}}^{(i)}(t)] \,\&\, 1 = bits[t]$ for $t \in \{0, 1, 2\}$, where $bits[t]$ is the correct logical value of input role t by the invariant at the source carriers. Comp (Alg. 2) then reads channel $\pi_{\mathrm{in}}^{(i)}(t)$ at every pixel as logical input role t, applies F, and writes output role t into channel $\pi_{\mathrm{out}}^{(i)}(t)$ at every pixel of the result image. At (row, col) specifically the bits read are exactly (c, x, y), the computation produces $F(c, x, y)$, and the three outputs land in channels $\pi_{\mathrm{out}}^{(i)}(0)$, $\pi_{\mathrm{out}}^{(i)}(1)$, $\pi_{\mathrm{out}}^{(i)}(2)$. Carol points the wire-table entries for G_i's three output wires at these three (image, channel) pairs, restoring the invariant.

Output extraction. For each primary output wire $w \in W_{\mathrm{out}}$, Bob reads the LSB of $\mathcal{W}[w]$'s image at (row, col) and the indicated channel. By the invariant after the last gate, this LSB is the correct logical output bit. Carol's pixel-uniform processing means the bit at (row, col) is computed identically to every other pixel's; only (row, col) carries the secret logical value, and Bob's secret position selects it. □

B. Empirical steganalysis

For each image size $s \in \{64, 128, 256\}$, 200 trials compare a fresh cover against a CRISP Comp output via the Pearson chi-square test on the LSB plane against $p_0 = 0.5$, which is χ^2_1 under the null. Table 6 reports mean χ^2, mean p-value, and false-positive rate at $\alpha = 0.05$.

Chi-square does not separate the two, which matches the structural argument of Section 6.6: Comp touches every pixel, so there is no localised modification footprint. Two caveats. These covers are synthetic and uniform, the easy case for any LSB test; on natural photographs the question is whether repair R1's re-randomised plane looks like a natural LSB plane, and it does not. Second, chi-square is a weak detector. Sample-pair analysis and learned detectors on real

Table 6: Chi-square detection on the LSB plane, 200 trials per row.

Image	Nat. χ^2	CRISP χ^2	Nat. p	CRISP p	Nat. FPR	CRISP FPR
64×64	1.13	1.11	0.49	0.50	0.075	0.070
128×128	1.12	0.84	0.50	0.55	0.060	0.040
256×256	1.15	1.04	0.49	0.52	0.065	0.060

image corpora are untested and we expect them to separate CRISP images from natural covers. That leaves Theorem 3 untouched, since it holds under known presence, but CRISP should not be called undetectable.