Algorithm 1 Arithmetic encoding

Require: Data sequence $D = \{d_1, d_2, \ldots, d_n\}$, symbol probabilities $\mathcal{P}$
Ensure: Encoded value E

Initialize $low \leftarrow 0$
Initialize $high \leftarrow 1$
Initialize $range \leftarrow high - low$
for each symbol d_i in D **do**
    $range \leftarrow high - low$
    $high \leftarrow low + range \times \mathrm{CDF}(d_i)$
    $low \leftarrow low + range \times \mathrm{CDF}(d_{i-1})$
end for
$E \leftarrow (low + high)/2$
return E

[redacted]

Algorithm 2 Arithmetic decoding

Require: Encoded value E, symbol probabilities $\mathcal{P}$, data length n
Ensure: Decoded sequence $\{D' = d'_1, d'_2, \ldots, d'_n\}$

Initialize $low \leftarrow 0$
Initialize $high \leftarrow 1$
Initialize $range \leftarrow high - low$
Initialize $encoded\_value \leftarrow E$
for $i \leftarrow 1$ to n **do**
    $range \leftarrow high - low$
    $value \leftarrow (encoded\_value - low)/range$
    Find the largest j such that $\mathrm{CDF}(d_j) \leq value$
    $d'_i \leftarrow d_j$
    $high \leftarrow low + range \times \mathrm{CDF}(d_j)$
    $low \leftarrow low + range \times \mathrm{CDF}(d_{j-1})$
end for
return S'

Algorithm 3 Data hiding algorithm

Require: Cover images I_1, I_2, secret data bits $m = \{b \in \{0,1\}^n\}$
Ensure: Marked images I_1', I_2'
Initialization:
Divide I_1 and I_2 into non-overlapping blocks of size 16×16
for each block pair (B_i^1, B_i^2) **do**

Algorithm 3 *Cont.*

    Encoding:
    for each pixel pair (x_k^1, x_k^2) **do**
        Extract upper bits to form codeword P
        Compute syndrome $S \leftarrow H \cdot P^T$
        $S' \leftarrow S \oplus (m_k, m_{k+1}, m_{k+2})$
        Store S' in encoded message M
    end for
    Compression:
    $D_{comp} \leftarrow A(M)$ ▷ Apply arithmetic encoding
    $L \leftarrow \text{len}(D_{comp})$
    Embedding (RDH EMD) method:
    for $k = 1$ to $L/2$ **do**
        Compute $f \leftarrow (x_k^1 + x_k^2 \cdot 2) \bmod 5$
        $d_{decimal}(k) = D_{comp}^{2r} \cdot 2^1 + D_{comp}^{2r+1} \cdot 2^0$ ▷ binary to decimal value
        Compute $pos = (d_{decimal}(k) - f) \bmod 5$
        if $pos < 0$ **then**
            $pos \leftarrow 5 - \text{abs}(pos)$
        end if
        if $pos == 1$ **then**
            $x_k^1 \leftarrow x_k^1 + 1$
        else if $pos == 2$ **then**
            $x_k^2 \leftarrow x_k^2 + 1$
        else if $pos == 3$ **then**
            $x_k^1 \leftarrow x_k^1 - 1$
            $x_k^2 \leftarrow x_k^2 + 2$
        else if $pos == 4$ **then**
            $x_k^1 \leftarrow x_k^1 + 1$
            $x_k^2 \leftarrow x_k^2 - 1$
        end if
    end for
    Store length L in pixels $\{x_{250}, x_{251}, \ldots, x_{256}\}$ of B_i^1
end for
Finalization:
Replace original blocks in I_1, I_2 with modified blocks B_i^1, B_i^2
return Marked images I_1', I_2'

Algorithm 4 Data extraction and image recovery algorithm

Require: Marked images I_1', I_2'
Ensure: Recovered original image O and hidden data m

Divide I_1' and I_2' into non-overlapping blocks of size 16×16
for each block pair (B_i^1, B_i^2) **do**
    Read pixel pairs $\{x_1^1, x_2^1, \ldots, x_{256}^1\}$ and $\{x_1^2, x_2^2, \ldots, x_{256}^2\}$
    Read binary values from pixel positions $(x_{250}, \ldots, x_{256})$ and convert to decimal length L
    for $k = 1$ to $L/2$ **do** ▷ Extract data using RDH EMD
        Compute $f \leftarrow (x_k^1 + x_k^2 \cdot 2) \bmod 5$
        $f_{binary} = \left(\left\lfloor \frac{f}{2} \right\rfloor \cdot 2\right) + (f \mod 2)$
        Assign $M(k) \leftarrow f_{binary}$
    end for
    for $j = 1$ to $L/2$ **do** ▷ Restore original pixel x_j
        Recover pixel value $p_j \leftarrow \left\lfloor \frac{x_j^1 \times x_j^2}{2} \right\rfloor$
        Replace corresponding pixels in blocks B_i^1, B_i^2
    end for
    Combine blocks to form images I_1, I_2
end for
Decompress extracted data using arithmetic decoding: $M_{decompressed} \leftarrow A_D(M)$
for each pixel pair (x_j^1, x_j^2) **do**
    Construct codeword $P_j = \{b_{j,7}^1, b_{j,6}^1, b_{j,5}^1, b_{j,4}^1, b_{j,7}^2, b_{j,6}^2, b_{j,5}^2\}$
    Convert $M_{j:j+2}$ to decimal $l \leftarrow b2d(M_{decompressed}^{j:j+2})$
    Flip bit at position l in P_j
    Extract hidden bits $m_{k:k+2} \leftarrow H(P_j \oplus e_l)$
end for
Combine all blocks to restore the original cover image O
return O, m