1 Fragile Watermark Preparation

The host image is divided into 2×4 size non-overlapping blocks (total blocks = TB). A random binary sequence W_{ran_2} is produced using K_3 (secret key-3) with the help of SFMT generator of length $6 \times$TB. The average block intensities of every 2×4 are converted into 8-bit binary representation. The 6 MSB (most significant bits) of these 8 bits are concatenated in a controlled randomized manner using K_4 (secret key-4) to generate recovery watermark data (W_{recov}). Now, every 6 bits from both sequences (i.e., W_{ran_2} and W_{recov}) are cascaded. Finally, the combined watermark $W_{fragile+recov}$ is obtained from this cascading. Figure 1 presents the process of generating the watermark $W_{fragile+recov}$.

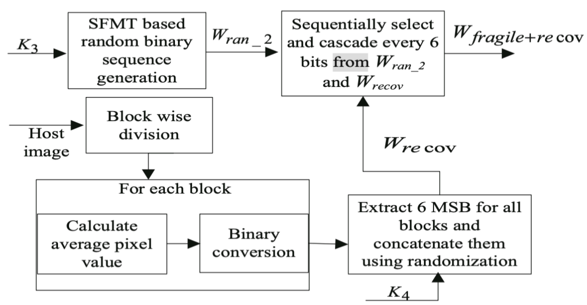

Figure 1: Fragile watermark preparation in order to preserve the recovery data along with the authentication data

3.2.2 Fragile Embedding

The following steps show the embedding of the watermark ($W_{\text{fragile+recov}}$) into the image (Watermarked$_{\text{r}}$).

- **Step-1** Divide the image (Watermarked$_{\text{r}}$) into 2×4 size of non-overlapping blocks. Then, sequentially select 12 bits of watermark data from $W_{\text{fragile+recov}}$ for each block.
- **Step-2** Choose the first block and consider that each column is representing a pixel unit (with 2 pixels). Thus, each 2×4 size block has four units (e.g., U1, U2, U3, and U4).
- **Step-3** Convert 12-bit watermark into 9-base number Wat_9 in such a way that it has four digits (e.g., $Wat_9 = d1d2d3d4$).
- **Step-4** Modify pixels of U1 using $d1$ as per the given steps.

– Compute digit F_{embed} as shown in Eq. (5). Here, P_k is the k^{th} pixel of unit U.

$$F_{\text{embed}} = \left(\sum_{k=1}^{n} 3^{k-1} P_k \right) \mod 3^n \quad \text{where} \quad n = 2$$

– Calculate x as given in Eq. (6).

$$x = \left(d - F_{\text{embed}} + \left\lfloor \frac{3^n - 1}{2} \right\rfloor \right) \mod 3^n$$

– Change x into x' by converting into 3-base number as $x' = y_1 y_2 \ldots y_n$, where y_i denotes the i^{th} digit of x' for $1 \le i \le n$. Next, get $x'' = z_1 z_2 \ldots z_n$, where $z_i = y_i - 1$.

– Add digits of x'' to the pixels of unit U to get the updated pixels as shown in Eq. (7).

$$P_{\text{new}_k} = P_k + z_i \quad \text{where} \quad \begin{cases} 1 \le k \le n \\ i = n - k + 1 \end{cases}$$

- Repeat the steps with U2, U3, and U4 to embed $d2$, $d3$, and $d4$, respectively.

- **Step-5** Repeat step 2, 3, and 4 for each block to get the dual watermarked image W_{img}.

2 Fragile Watermark Extraction (Tamper Check)

The following steps show the authentication process to check the tampering:

1. Divide the attacked watermarked image into 2×4 size blocks and extract the four digits (F_{ext}) using Eq. (5) from each block.

2. Convert the four-digit 9-base number (i.e., $F_{ext\_1}F_{ext\_2}F_{ext\_3}F_{ext\_4}$) into binary (i.e., 12-bit size). Similarly, extract 12-bit numbers from each block.

3. Get EW_{ran_2} by concatenating initial 6-bits concerning each block. Likewise, get EW_{recov} by cascading the last 6 bits related to each block.

4. Generate binary sequence W_{ran_2} using K_3-based SFMT process and Eq. (1). Further, compare the corresponding bits of EW_{ran_2} and W_{ran_2} to authenticate the image. If the bits are different, then the concerning block is marked as tampered.

5. Apply the block-neighborhood approach to smoothen the result of Step 4. For smoothing, eight neighborhood blocks (except corner positions) of each block are considered. If the majority of the blocks (out of these nine blocks) are tampered/non-tampered, the block is also marked as tampered/non-tampered.