IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 2, 2019, accepted December 30, 2019, date of publication January 3, 2020, date of current version January 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963711

Structural Obfuscation and Crypto-Steganography-Based Secured JPEG Compression Hardware for Medical Imaging Systems

ANIRBAN SENGUPTA, (Senior Member, IEEE), AND MAHENDRA RATHOR, (Member, IEEE)
IIT Indore, Indore 453552, India

Corresponding author: Anirban Sengupta (asengupt@iiti.ac.in)

This work was supported in part by the Digital India Corporation (Media Lab Asia), Ministry of Electronics and IT.

ABSTRACT In modern healthcare technology involving diagnosis through medical imaging systems, compression and data transmission play a pivotal role. Medical imaging systems play an indispensable role in several medical applications where camera/scanners generate compressed images about a patient's internal organ and may further transmit it over the internet for remote diagnosis. However, tampered or corrupted compressed medical images may result in wrong diagnosis of diseases leading to fatal consequences. This paper aims to secure the underlying JPEG compression processor used in medical imaging systems that generates the compressed medical images for diagnosis. The proposed work targets to secure the JPEG compression processor against well-acknowledged threats such as counterfeiting/cloning and Trojan insertion using double line of defense through integration of robust structural obfuscation and hardware steganography. The second line of defense incorporates stego-key based hardware steganography that augments the following: non-linear bit manipulation using S-box (confusion property), diffusion property, alphabetic encryption, alphabet substitution, byte concatenation mode, bit-encoding (converting into stego-constraints) and embedding constraints. The results of the proposed approach achieve robust security in terms of significant strength of obfuscation, strong stego-key size (775 bits for JPEG compression processor and 610 bits for JPEG DCT core) and probability of coincidence of 9.89e-8, at nominal design cost.

INDEX TERMS Medical imaging, JPEG compression processor, double line of defense, threats, hardware steganography, obfuscation.

I. INTRODUCTION

In the current era of healthcare technology, the roles of hardware chips and internet are very crucial. The electronic integrated circuits (ICs) and internet technology have eased and fastened the diagnosis and treatment of critical diseases. More explicitly, in the medical applications such as computed tomography (CT) scan, magnetic resonance imaging (MRI) scan etc., the healthcare professionals acquire the medical images of the patient's internal organs for diagnosis. The imaging modalities (medical instruments) such as CT scanner and MRI scanner generate large size of digital image data [1]. These digital medical images are stored/processed locally and subsequently transmitted to healthcare professionals through internet. However, the amount of space/bandwidth required to store/transmit the images can be prohibitively large. For example, the size of each slice of CT abdomen images is 512×512 pixels, where each pixel is of 16 bits. In addition, the entire data set of CT abdomen images contains 200 to 400 images resulting into $\sim$150 MB of data [2], [3]. Therefore, the images are required to be stored/ transmitted in the compressed form. Additionally, it also improves the efficiency of medical data transmission over internet for tele-radiology and tele-pathology [1].

Lossy compression of medical images can be used within acceptable range of compression ratio which differs for different modalities and body parts [1], [4]–[6]. Alternatively, hybrid compression technique may also be used. Diagnostically important regions i.e. Region of Interest (ROI) may require high image quality. In such cases, high compression

The associate editor coordinating the review of this manuscript and approving it for publication was Chunsheng Zhu.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

with good quality in diagnostically important regions may be used along with lossy compression in other regions [2], [3].

A. TARGET THREAT MODEL

Due to emerging threats at the hardware level, corruption of compressed medical images generated from camera, scanners are significantly surging. For example, corruption of compressed medical images through attacks in the compression processor in the imaging systems can manifest in one or more of the following forms: alteration (tampering) of acceptable compression ratio, affecting the diagnostically important regions of interest by damaging the image quality etc. This is because the underlying processor responsible for medical image compression may be compromised due to hardware threats such as reverse engineering (RE) causing Trojan insertion [7], piracy and counterfeiting etc. [8]– [10].

B. MOTIVATION OF THE PROPOSED APPROACH

To generate the compressed medical images/data through medical imaging systems (such as camera, scanners etc.) in its authentic form, the underlying JPEG compressor-decompressor (CODEC) processor [1] needs to be reliably secured against hardware threats. This is because an unsecured JPEG processor may contain malicious logic (such as Trojans) and may be counterfeited/cloned. The Trojan infected/counterfeited /cloned hardware is not trustworthy, hence may generate tampered/corrupted/altered compressed medical data. The generated corrupted medical image data results into wrong diagnosis of diseases. Therefore, the security of the underlying processor of the camera/micro-camera used in medical instruments needs to be ensured against aforementioned threats. Robust security can be ensured by taking both preventive and detective measures against such potential threats. **Therefore, the security (prevention and detection) of this JPEG CODEC processor of camera/scanner against hardware threats such as Trojan insertion, counterfeiting/cloning is mandatory to produce genuine medical data**.

II. RELATED WORK

There exists some approaches [11], [13], [20] in the literature that secures hardware against RE/Trojan insertion using structural obfuscation. In related prior work [11], [20], structural obfuscation has been applied on discrete cosine transform (DCT) core which is used underneath JPEG compression processor. However, [11], [20] do not perform structural obfuscation on entire JPEG CODEC processor and do not provide detective control against counterfeiting/cloning. Further in [13], structural obfuscation based security has been performed on JPEG CODEC hardware. Reference [13] performed the structural obfuscation based on tree height transformation (THT), however does not incorporate hardware steganography. Hence, this approach provides countermeasure against RE, however fails to detect in case of pirated JPEG processors. Moreover, there also remains a possibility that the obfuscated JPEG design can be de-obfuscated by a potential attacker. Once de-obfuscated, the JPEG design can be tampered. As a solution to this problem, the proposed approach embeds the hardware steganography as 2^{nd} line of defense into the obfuscated JPEG processor to facilitate the detection of counterfeiting/cloning. Additionally, [21] employed entropy based single phase hardware steganography and [16], [17], [22], [23] employed watermarking technique to secure the designs against counterfeiting/cloning. However, [16], [17], [21]–[23] provided only detective control against counterfeiting /cloning and did not ensure the preventive security against RE (causing Trojan insertion attack). Proposed approach is the first work in the literature that secures JPEG CODEC processor by employing structural obfuscation followed by proposed crypto-based dual phase hardware steganography (double line of defense) against aforementioned threats. Here, it is worth noting that the proposed approach embeds digital evidence in JPEG compression hardware as 2^{nd} line of defense. Hence, the focus of proposed approach is not on embedding/hiding data in JPEG images. Therefore, data hiding schemes have not been discussed in the paper as it does not apply to the context of the problem.

Novel contributions of the proposed work are as follows:

- The proposed work aims to secure the underlying JPEG compression processor used in medical imaging systems that generates the compressed medical images for diagnosis.
- The paper proposes double line of defense by integrating structural obfuscation and hardware steganography together to secure JPEG CODEC processor against Trojan insertion, counterfeiting/cloning.
- The paper proposes a novel crypto-based dual phase steganography that acts as a second line of defense (detective control against cloning/counterfeiting). This enables separation of counterfeited/cloned JPEG CODEC ICs from being used in the design/manufacturing process of the medical imaging systems.

III. PROPOSED METHODOLOGY

A. OVERVIEW

Structural obfuscation and hardware steganography can be integrated together to provide double line of defense to secure JPEG CODEC processor against Trojan insertion, counterfeiting/cloning. Structural obfuscation provides preventive control by obscuring the architecture (or structural topology of the circuit) of the design in order to make it unobvious for an attacker in understanding, thus making it harder to reverse engineer or identify the functionality of the design [11]–[13]. This prevents an attacker from secretly inserting malicious logic into the underlying design as well as secures against cloning/counterfeiting. Further, hardware steganography acts as the second line of defense by providing detective control, in case a potential attacker is able to crack the structural obfuscation using advanced algorithms

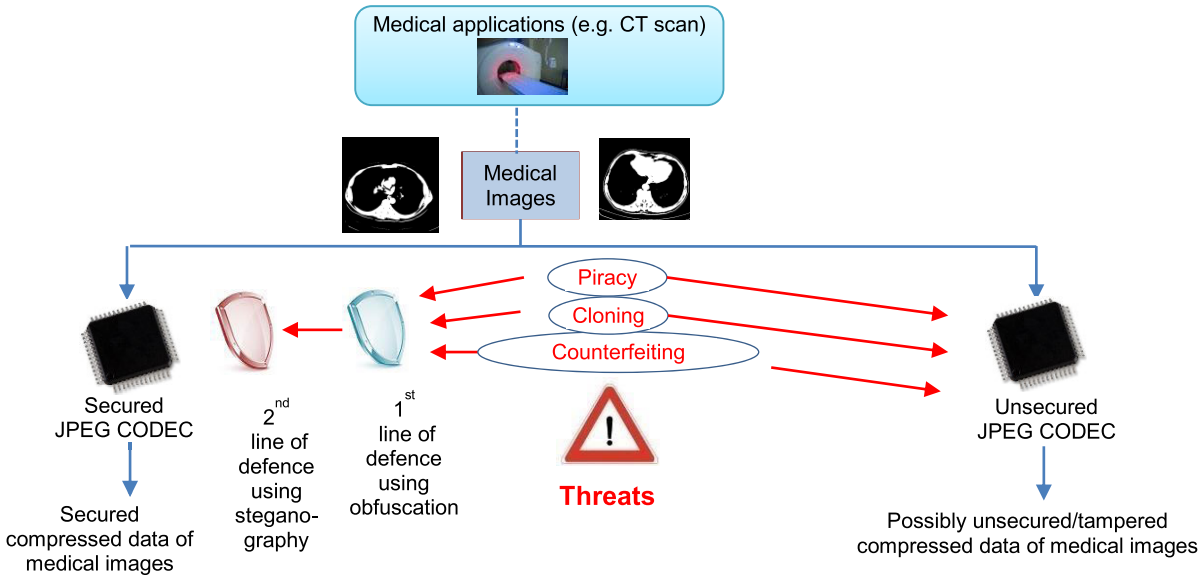


FIGURE 1. Thematic representation of securing medical images against external hardware threats.

to realize his/her malicious intents. Therefore, there is need of a double line of defense that is capable to both prevent and detect a tampered hardware. Hardware steganography as the second line of defense provides detection and isolation of the counterfeited and cloned IPs from genuine ones during authentication process. Fig. 1 shows the thematic representation of typical attack scenario on JPEG CODEC processor used in medical imaging. Further, Fig. 2 depicts the targeted hardware threats and protection scenario against them using proposed approach. as shown two classes of hardware threats viz. RE (resulting into Trojan insertion) and piracy (counterfeiting/cloning) for JPEG processor have been countered through proposed algorithm using structural obfuscation and crypto-hardware steganography. The countermeasure summaries are also enunciated. The scope of proposed approach does not cover the data hiding schemes.

The proposed approach aims to secure the underlying JPEG CODEC processor responsible for image compression in medical imaging systems. The use of authorized and secured JPEG CODEC processor ensures that the medical images are stored in the genuine form and thus leads to accurate diagnosis by healthcare professionals. The proposed approach provides enhanced security by embedding crypto-based dual phase steganography information into an obfuscated JPEG compression processor as a 2^{nd} line of defenses. The 2^{nd} line of defense enables the detection of piracy, counterfeiting and cloning and thus filters out the unauthorized ICs.

The proposed methodology secures the JPEG CODEC processor using double-line of defense. The inputs to the proposed methodology are as follows: (a) JPEG CODEC algorithm in the form of C-code/mathematical function or its equivalent data flow graph (DFG) [13] (b) resource constraints (c) module library (d) stego-keys. The overview

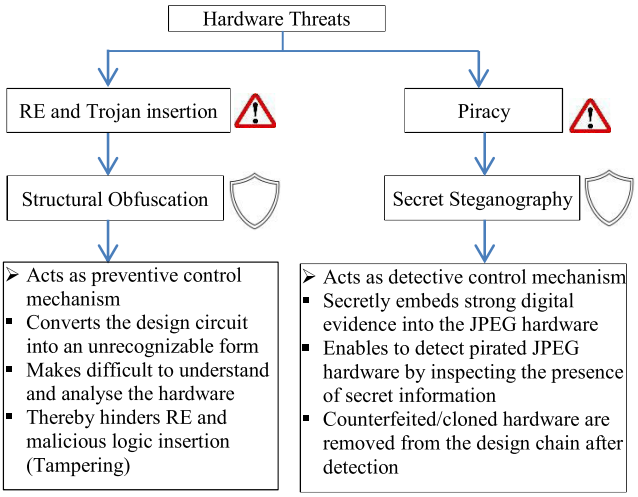


FIGURE 2. Overview of hardware threats and protection scenarios using proposed approach.

of the proposed approach is shown in Fig. 3. The upper half of the figure shows the process of securing JPEG CODEC using double line of defense. As shown in the figure, the DFG representing JPEG compression algorithm is structurally obfuscated based on tree height transformation (THT) based obfuscation. Here, structural obfuscation acts as 1^{st} line of defense against malicious logic insertion (such as Trojan), counterfeiting/cloning by obscuring the architecture (or structural topology) of the design in order to make it harder for an attacker to reverse engineer or identify the functionality of the design. Further, the 2^{nd} line of defense is deployed by embedding hardware steganography into the structurally obfuscated JPEG compression processor design. The proposed crypto-based dual-phase steganography encoder generates stego-constraints and embeds them

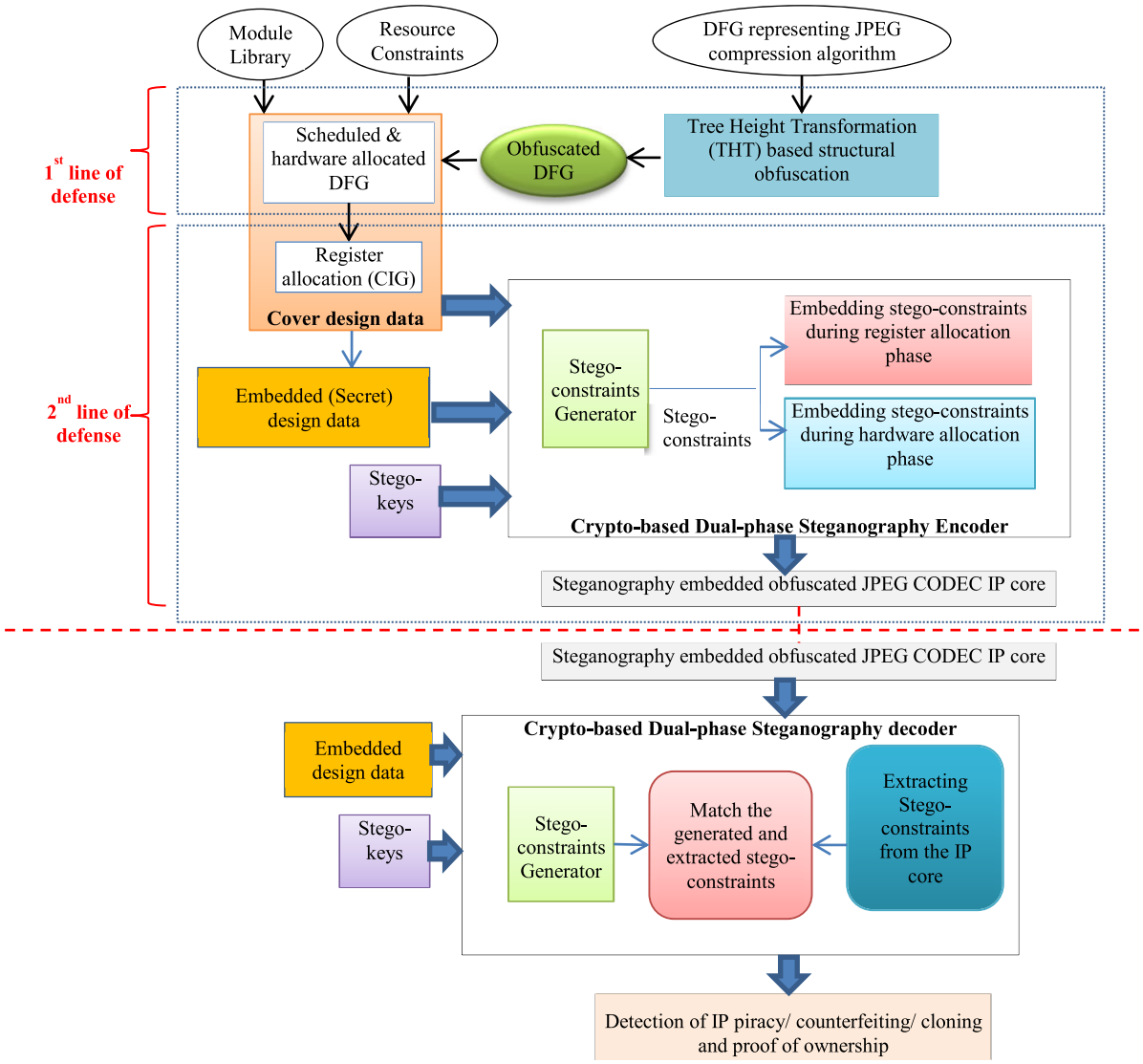


FIGURE 3. Proposed double line defense using structural obfuscation and crypto-based dual-phase steganography to secure JPEG CODEC processor used for compression of medical images.

into the obfuscated JPEG design in the form of secret information. The steganography information is embedded covertly into the two distinct phases of high level synthesis (HLS) [14], [15] viz. register allocation phase and hardware allocation phase [16]. The process of generating secret stego-information cooperatively leverages the following: (a) non-linear bit manipulation (Shannon's confusion property) (b) row and column diffusion (Shannon's diffusion property) (c) TRIFID cipher (achieves both confusion and diffusion). Thus the proposed approach is referred as crypto-based dual-phase hardware steganography. The output of the proposed methodology is a steganography embedded obfuscated JPEG compression Processor.

Besides, the lower half of the Fig. 3 shows the steganography decoder system which helps in detecting piracy, counterfeiting and cloning. As shown in figure, stego-constraints (concealed stego-information) are extracted from the design under test. This extracted stego-information is matched with the stego-constraints generated using the designers provided secret design data (embedded design data) and stego-keys. If all stego-constraints match then the JPEG compression processor is authentic and suitable for using in medical imaging systems. However, if either the stego-constraints do not match or a different brand carries the same stego-information, then the design is considered to be counterfeited/cloned, hence discarded.

It is worth noting that the proposed approach does not apply for protecting image contents. Instead, the proposed approach protects the JPEG processor responsible for generating compressed image data against tampering attacks (such as inserting malicious logic which is safeguarded using obfuscation. Counterfeited/cloned hardware could be tampered hardware which is safeguarded by steganography based detective control).

B. DETAILS OF PROPOSED APPROACH

As shown in the Fig. 3, the process of securing JPEG compression processor is accomplished by deploying 1st line of defense followed by 2nd line of defense.

1st Line of Defense: The 1st line of defense is deployed by performing tree height transformation (THT) based structural obfuscation on DFG of JPEG compression processor. In THT based structural obfuscation [11], [13], some sequentially executing operations are forced to execute as parallel sub-computations, thus incorporating the structural changes by altering the data dependency of operations. This would result into structural obfuscation in the form of changes in functional units such as adders, multipliers etc., storage elements (register count, latches etc.) and interconnectivity of resources, multiplexer and de-multiplexer inputs/outputs, without affecting the func-tionality. Thus would **affect the datapath architecture and controller logic of the design** without changing functionality. The resultant effect would be significant change in gate-level netlist with no change in functional behavior. The structurally obfuscated design becomes unobvious for an attacker to identify the functionality. Thus, the structure of the design remains concealed for the attacker and he/she fails to insert malicious logic (Trojan) and counterfeit/clone the hardware. Thereby, structural obfuscation acts as 1st line of defense.

2nd Line of Defense: The scheduled and allocated design based on resource constraints and module library is obtained from the obfuscated DFG of JPEG compression algorithm earlier. This is used as input for embedding hardware steganography, as 2nd line of defense. The process of embedding steganography information in the form of stego-constraints is accomplished through crypto-based dual phase steganography encoder system. The process is explained in the following sub-sections:

1) INPUTS/OUTPUT OF THE PROPOSED HARDWARE STEGANOGRAPHY ENCODER: 2ND LINE OF DEFENSE

The encoder system uses following three primary inputs to generate steganography embedded obfuscated JPEG compression processor:

a: COVER DESIGN DATA

For the proposed steganography approach, following two forms of the JPEG design are exploited as cover data: (a) scheduled and hardware allocated obfuscated DFG of JPEG compression processor (b) colored interval graph (CIG) [16] of the design, where nodes represent storage variables of the design, colors indicate different registers and edges between the nodes represent overlapping life-time of respective nodes. A CIG is a graphical representation used to show the assignment of all storage variables (Vj) in the JPEG design to the minimum possible registers/colors [10], [16].

To obtain the cover design data, the obfuscated DFG of the JPEG compression is scheduled and hardware allocated based on designer selected resource constraints and module library. Further, a CIG or register allocation is obtained using the scheduled DFG. Thus obtained CIG/register allocation is used to embed phase-1 of steganography (the stego-constraints are embedded into the CIG during register allocation phase [16] of HLS). The forced execution of some storage variables through different registers (instead of their default registers) shows the embedded stego-constraints post phase-1 steganography. Further, another form of the cover design (scheduled and hardware allocated DFG of JPEG compression) is used to embed stego-constraints during hardware allocation phase [10] of HLS (phase-2 of steganography). The reallocation of functional units (FU) hardware shows the embedded stego-constraints in phase-2 steganography.

b: EMBEDDED DESIGN DATA

This is the secret design data used to generate stego-constraints to be embedded. This secret design data is extracted from the corresponding CIG/ register allocation of JPEG compression.

c: Stego-Keys

Stego-key is very important part of the steganography process, because it controls the amount of steganography to be embedded into the JPEG processor design. In the proposed approach, total five stego-keys are used at different stages of stego-constraints generation process.

The output of the steganography encoder system is a steganography embedded obfuscated JPEG compression processor/intellectual property (IP) core.

2) DETAILS OF STEGANOGRAPHY ENCODER SYSTEM

Based on the stego-keys value and embedded design data (secret design data), the steganography encoder system generates stego-constraints and embeds them into the cover design data. Fig. 4 shows the detailed process of generating and embedding stego-constraints to obtain a steganography embedded obfuscated JPEG compression Processor. Following are the steps:

1. Embedded (Secret) Design Data: A secret design data is extracted from the CIG/register allocation. The secret design data is represented by a set 'S' comprising of elements indicating 'indices value (i, k) of storage variable pair (Vi, Vk) assigned to same register/color in the CIG. Thus, each element of the set is a pair of two digits. In order to represent each digit in the set in the hexadecimal form, '15' is subtracted from those digits whose value is greater than 15. The set 'S' in the current form is used as embedded (secret) design data for stego-constraints generation.

2. Stego-Constraints Generation: Based on the secret design data and stego-keys, stego-constraints are generated for embedding in the cover design data. The types of stego-key used and its corresponding modes are shown in Fig. 5 (Note: the modes are extendable as per designer's choice. The number of modes shown here are arbitrary). As shown in the Fig. 4, following steps are performed to generate the stego-constraints:

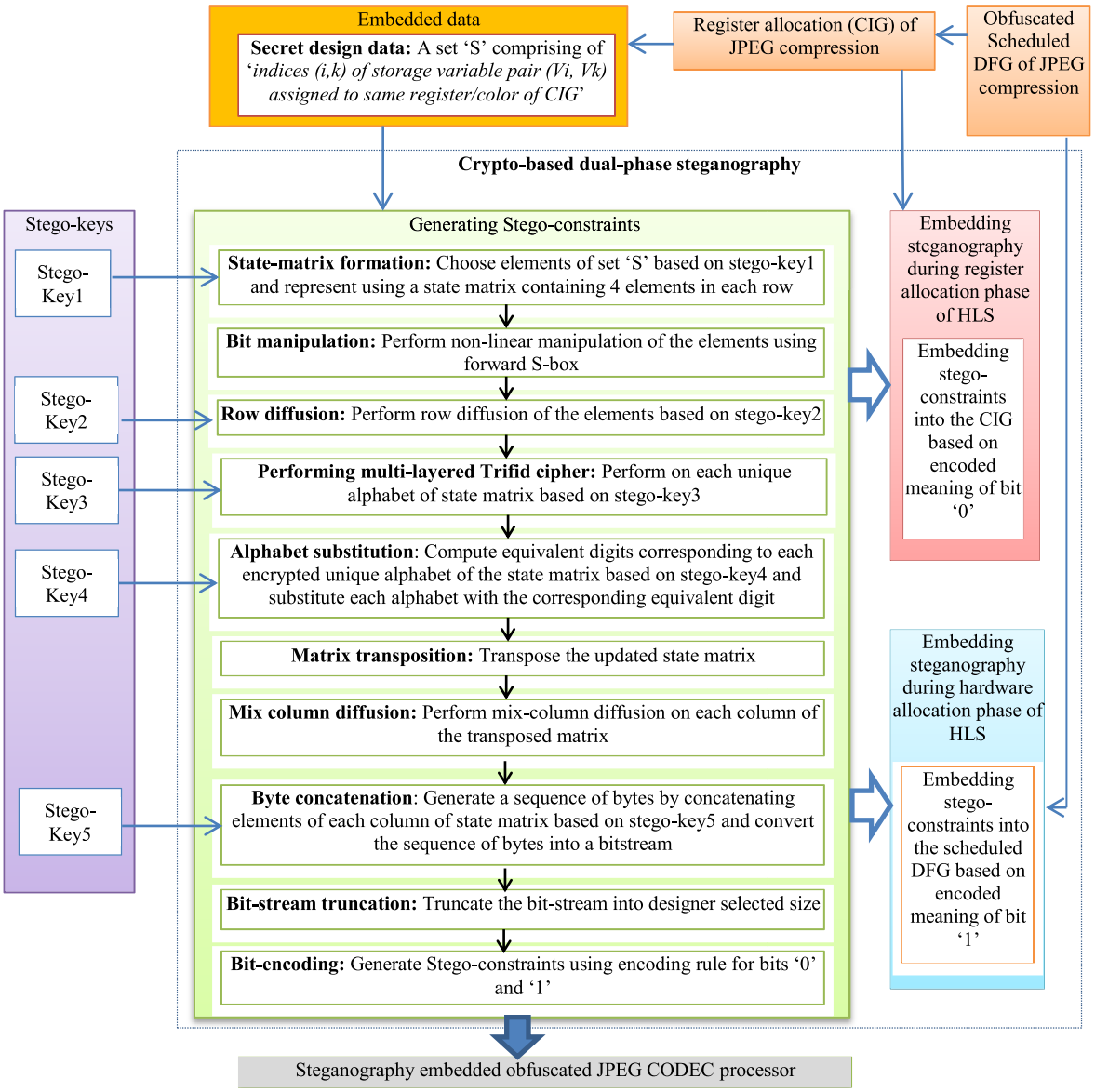


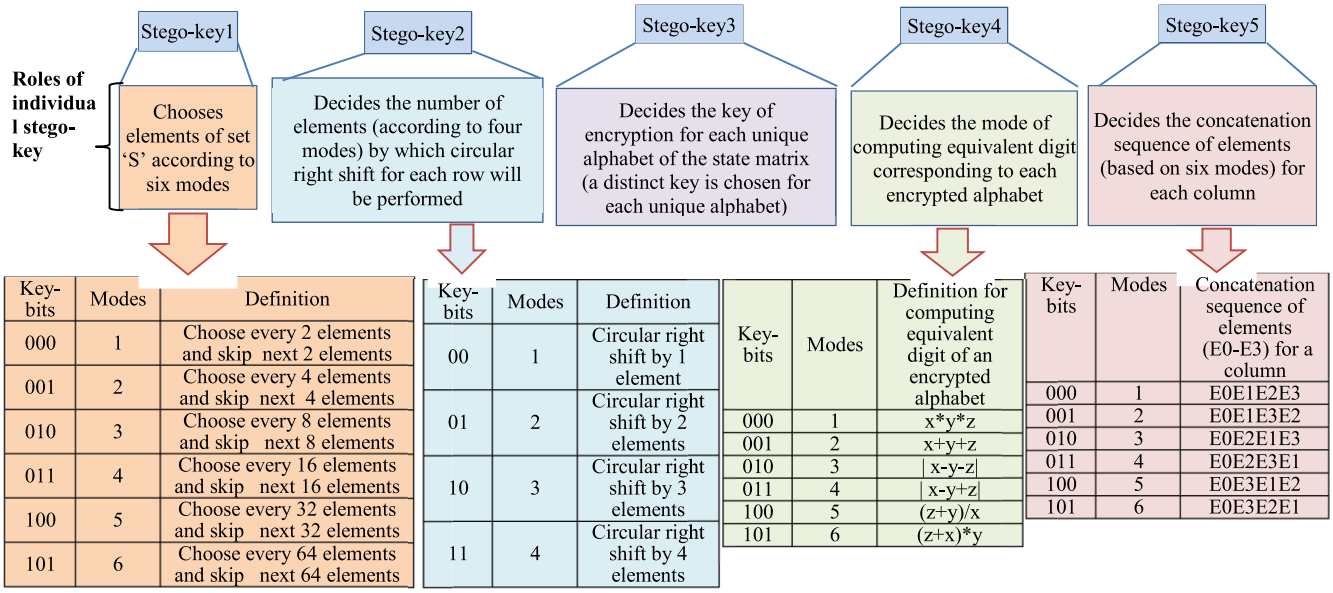
FIGURE 4. Details of 2nd line of defense: Proposed crypto-based dual-phase steganography.

(a) State-matrix formation: Based on stego-key1, some elements of set 'S' (representing secret design data) are chosen to form a state matrix, containing four elements in each row. The role of stego-key1 in choosing the elements from set 'S' is shown in Fig. 5. As shown in the figure, the size of stego-key1 is 3 bits. Thus combination of three bits decides the mode of choosing the elements from set 'S'. Total six modes for choosing elements are shown in Fig. 5. The selection of appropriate mode depends on the JPEG compression processor designer.

(b) Bit-manipulation: The nonlinear bit-manipulation (byte substitution) is performed on each element of the matrix using forward S-box. The bit-manipulation is performed to obscure the relationship between the key and the final stego-constraints generated later (based on Shannon's property of confusion).

(c) Row diffusion: Post bit-manipulation, row diffusion is performed in the matrix to obscure the relationship between input secret design data and stego-constraints generated later (based on Shannon's property of diffusion). The diffusion is performed based on stego-key2. The stego-key2 decides the number of elements by which circular right shift in each row will be performed. The size of stego-key2 depends on the number of rows in the state matrix. If there are 'R' number of rows in the matrix, then the size of stego-key2 is 2*R bits. This is because, for each row (containing 4 elements), combination of two bits decide the mode of row diffusion. Fig. 5 shows the role of stego-key2 and definition of different modes of diffusion for a row of the state matrix.

(d) Performing Multi-layered Trifid cipher based encryption: Proposed algorithm uses Trifid cipher to provide following properties (i) fractionation (ii) transposition



Key-bits	Modes	Definition
000	1	Choose every 2 elements and skip next 2 elements
001	2	Choose every 4 elements and skip next 4 elements
010	3	Choose every 8 elements and skip next 8 elements
011	4	Choose every 16 elements and skip next 16 elements
100	5	Choose every 32 elements and skip next 32 elements
101	6	Choose every 64 elements and skip next 64 elements

Key-bits	Modes	Definition
00	1	Circular right shift by 1 element
01	2	Circular right shift by 2 elements
10	3	Circular right shift by 3 elements
11	4	Circular right shift by 4 elements

Key-bits	Modes	Definition for computing equivalent digit of an encrypted alphabet
000	1	x*y*z
001	2	x+y+z
010	3	| x-y-z|
011	4	| x-y+z|
100	5	(z+y)/x
101	6	(z+x)*y

Key-bits	Modes	Concatenation sequence of elements (E0-E3) for a column
000	1	E0E1E2E3
001	2	E0E1E3E2
010	3	E0E2E1E3
011	4	E0E2E3E1
100	5	E0E3E1E2
101	6	E0E3E2E1

FIGURE 5. Roles of five stego-keys and definition of their different modes.

(iii) substitution. These properties achieve certain amount of confusion and diffusion which help in obscuring the relationship of the generated stego-constraints with the input secret design data and stego-keys. Multi-layered Trifid cipher is performed on each unique alphabet of state matrix based on stego-key3. The stego-key3 decides the key of encryption for each unique alphabet of the state matrix. For each unique alphabet, the encryption key is 27 characters long. Since, there are 27! permutations of 27 characters, therefore + ($\log_2(27!)$ + bits are required to represent the key for an alphabet. Thus, total size of key to encipher all unique alphabets is = (# of unique alphabets)* $\lceil(\log_2(27!)\rceil$. This is because, a distinct key is chosen for each unique alphabet. The process of performing Trifid cipher is illustrated using an example in the next sub-section.

(e) Alphabet substitution: For each encrypted alphabet, equivalent digit is computed based on stego-key4. The size of the stego-key4 = (# of unique alphabets)*$\lceil(\log_2$(# modes for computing digit equivalent of unique alphabet)$\rceil$. The role of stego-key4 and the definition of each mode for computing digit equivalent of unique alphabets are shown in Fig. 5. Further, the alphabets of state matrix are substituted with the corresponding equivalent digit obtained.

(f) Matrix transposition: The updated state matrix is transposed.

(g) Mix-column diffusion: It is performed on each column of the transposed matrix to introduce Shannon's property of diffusion. For each column of the transposed matrix, the mix column diffusion is performed by using a circulant MDS (Maximum Distance Separable) matrix as used in advanced encryption standard (AES). The process of performing mix-column diffusion is illustrated using an example in the next sub-section.

(h) Byte-concatenation: Once mix-column diffusion is performed, elements of each column (byte) are concatenated to generate a sequence of elements. The concatenation of each column elements is performed based on stego-key5. The size of stego-key5 = (# of columns in the transposed matrix)*$\lceil$ ($\log_2$(# of modes for concatenation) $\rceil$. For each column, six modes shown in the Fig. 5 decide the concatenation sequence of elements. The role of stego-key5 and the definition of each mode are shown in Fig. 5. Once all elements (bytes) of the state matrix are concatenated based on stego-key5, the sequence of elements is converted into a bit-stream.

(i) Bit-stream truncation: The bit-stream obtained in the previous step is truncated to a designer's selected size/strength.

(j) Bit-encoding: Once the designer selected size of bit-stream is obtained, bit '0' and bit '1' are encoded to convert into respective stego-constraints. Encoding of bit '0' and bit '1' representing the stego-constraints for embedding is shown in Table 1.

3. Stego-Constraints Implantation: Bit '0's are implanted into the design during register allocation phase and bit '1's are implanted during hardware allocation/scheduling phase of HLS. Embedding bit '0' and bit '1' as stego-constraints into the design generates steganography embedded JPEG compression processor.

$$\begin{aligned}\text{Total key size of the proposed hardware steganography} &= \text{key size of stego-key1} + \text{key size of stego-key2} \\ &+ \text{key size of stego-key3} + \text{key size of stego-key4} \\ &+ \text{key size of stego-key5} = 3\text{bits} + 2^{*}\text{R} \\ &+ (\#\text{of unique alphabets})^{*}\lceil(\log_2(27!)\rceil \\ &+ (\#\text{of unique alphabets}) * \lceil(\log_2(\#\text{modes for} \\ &\times \text{computing digit equivalent of unique alphabet})\rceil\end{aligned}$$

TABLE 1. Encoding of bit '0' and '1' in JPEG compression design.

Bit	Encoded meaning
0	Embed an edge between node pair (even, even) of CIG (during register allocation of HLS) of JPEG compression
1	In the JPEG compression scheduling/allocation, odd operations are assigned to FU of vendor type 1 (U1) and even operations are assigned to FU of vendor type 2 (U2) (during functional unit (FU) allocation phase of HLS)

Pseudo code of proposed double line of defense

```
1st line of defense:
Inputs: DFG of JPEG compression processor, module library,
        resource constraints
Output: Obfuscated scheduled DFG
Algorithm:
        Read DFG;
        perform_THT( ){
          traverse all operations;
          read data dependency of each operation;
          if operations of same type are executing sequentially,
          then execute them as parallel sub-computations while
          altering data dependency without changing functionality;
          return obfuscated DFG; }
        Read module library
        Read constraints file;
        architectural_synthesis( )
end algorithm;

2nd line of defense:
Inputs: Obfuscated scheduled DFG, Stego-key1 to Stego-key5
Output: Steganography embedded obfuscated JPEG processor
Algorithm:
        Read obfuscated scheduled DFG;
        create_CIG( )
        extract_secret_design_data( )
        state_matrix_formation( )
        bit_manipulation( )
        row_diffusion( )
        trifid_cipher( )
        alphabet_substitution( )
        matrix_transposition( )
        column_diffusion( )
        byte_concatenation( )
        bitstream_truncation( )
        bit_encoding( )
        embedding_bit_0( )
        embedding_bit_1( )
        datapath_controller_synthesis_JPEG_processor( )
end algorithm;
```

FIGURE 6. Pseudo code of proposed double line of defense.

$$+(\#\text{of columns in the transposed matrix})^{*} \lceil(\log_2(\#\text{of modes for concatenation})\rceil \quad (1)$$

Pseudo code of the proposed double line of defense is shown in Fig. 6.

C. DEMONSTRATION OF EMBEDDING HARDWARE STEGANOGRAPHY ON DCT CORE USED IN JPEG COMPRESSION

The JPEG compression uses discrete cosine transform (DCT) underneath which is responsible for transforming the image information into frequency domain [3]. A demonstration of securing DCT core by embedding proposed hardware steganography is shown using the following steps (this subsection only illustrates the process of embedding proposed steganography):

TABLE 2. Allocation of storage variables to register /colors in 8-point DCT before implanting steganography.

T	Pink	Indigo	Violet	Green	Yellow	Orange	Red	Black
0	V0	V1	V2	V3	V4	V5	V6	V7
1	V8	V9	V10	V11	V4	V5	V6	V7
2	V16	--	V10	V11	V12	V13	V14	V15
3	V17	--	--	V11	V12	V13	V14	V15
4	V18	--	--	--	V12	V13	V14	V15
5	V19	--	--	--	--	V13	V14	V15
6	V20	--	--	--	--	--	V14	V15
7	V21	--	--	--	--	--	--	V15
8	V22	--	--	--	--	--	--	--

- 8-point DCT is scheduled and hardware allocated based on say, resource constraints (1+,4*), maximum two instances of vendor type-1 and one instance of vendor type-2 for each functional unit type (e.g. in a control step, maximum two multipliers/adders of vendor type-1 and one multiplier/adder of vendor type-2 are allocated). The scheduled and hardware allocated DFG is shown in Fig. 7.
- A CIG is constructed form scheduled DFG as shown in Fig. 8. The corresponding register allocation (before implanting steganography) is shown in Table 2. The scheduled DFG and CIG/register allocation act as cover design data for embedding proposed hardware steganography.
- A secret design data extracted from CIG is represented by a set 'S' as follows:
  S = {(0,8), (0,16), (0,17), (0,18), (0,19), (0,20), (0,21), (0,22), (8,16), (8,17), (8,18), (8,19), (8,20), (8,21), (8,22), (16,17), (16,18), (16,19), (16,20), (16,21), (16,22), (17,18), (17,19), (17,20), (17,21), (17,22), (18,19), (18,20), (18,21), (18,22), (19,20), (19,21), (19,22), (20,21), (20,22), (21,22), (1,9), (2,10), (3,11), (4,12), (5,13), (6,14), (7,15)}
  Each element in the set represents the index value (i, k) of node pairs (Vi,Vk) of same color in the CIG.
- Further, digit '15' is subtracted from those digits of the set whose value is greater than 15. This is done to get each digit in the rage of 0-15 to represent in the corresponding hexadecimal form as shown: S = {(0,8), (0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (0,7), (8,1), (8,2), (8,3), (8,4), (8,5), (8,6), (8,7), (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (2,3), (2,4), (2,5), (2,6), (2,7), (3,4), (3,5), (3,6), (3,7), (4,5), (4,6), (4,7), (5,6), (5,7), (6,7), (1,9), (2,A), (3,B), (4,C), (5,D), (6,E), (7,F)}
- Based on stego-key1, a state-matrix is constructed using set 'S'. Suppose, the stego-key1 is "001". Therefore, mode-2 (choose 4 elements and skip 4 elements) is selected (shown earlier in Fig. 5). Based on the selected mode, the state-matrix containing four elements in each row is shown in Fig. 9(a). Note: Since last set of chosen

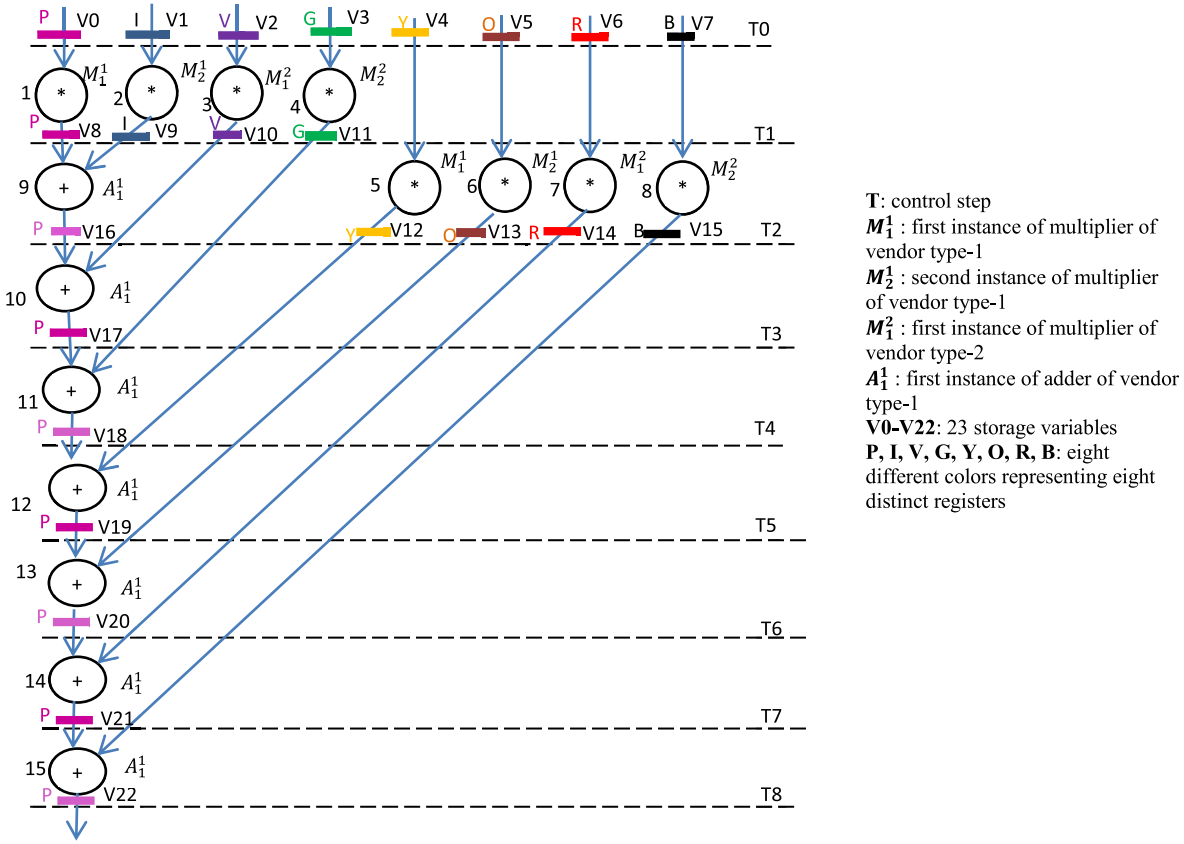


FIGURE 7. Scheduled and hardware allocated 8-point DCT using 1 (+) and 4 (∗) before implanting steganography.

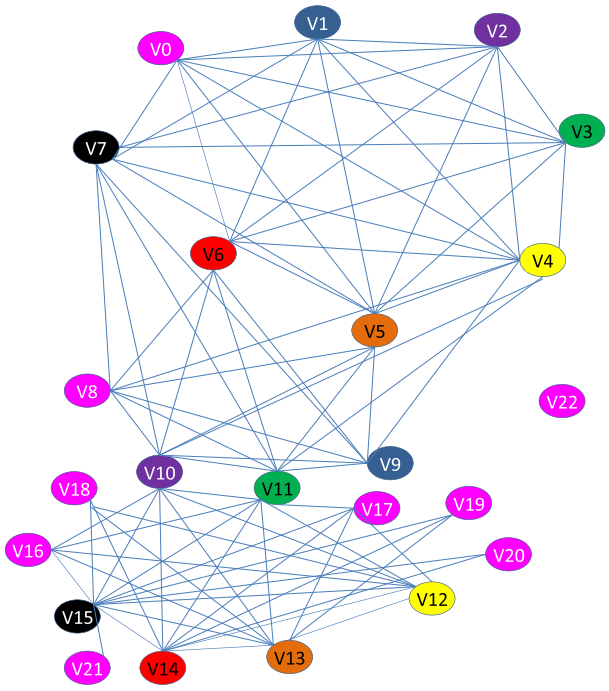


FIGURE 8. CIG of 8-point DCT before implanting hardware steganography.

elements {(5,D), (6,E), (7,F)} is not a multiple of four, thus is discarded.

- The modified state-matrix after performing bit-manipulation using forward S-box is shown in Fig. 9(b).
- Based on stego-key2, row diffusion is performed in the state-matrix. For the chosen key, mode of diffusion for each row has been shown in Fig. 5. The number of rows in the matrix is R = 5, therefore the size of stego-key2 is 2*R = 10 bits. Let's assume the stego-key2 is "01 00 10 00 11". Here, groups of two bits from left to right represent the key-bits for row-1 to row-5 respectively. Since, the key bits for first row is "01", therefore circular right shift by two elements is performed for this row. Similarly, diffusion is performed for each row based on the corresponding key bits (referring Fig. 5). Post row diffusion, the modified state-matrix is shown in Fig. 9(c).
- In this step, TRIFID cipher is performed on each unique alphabet of the state-matrix. The unique alphabets in the matrix are: A, B, C, D, E and F. Since, the key for each alphabet is 27 characters long, therefore the 27 characters are divided in three square matrices of size 3 × 3. The output of TRIFID cipher for each alphabet is represented by a state "xyz", where 'x', 'y' and 'z' denote the row number, column number and square matrix number respectively for the corresponding alphabet. The process of obtaining the state "xyz" for each unique alphabet is as follows:

08	01	02	03
81	82	83	84
13	14	15	16
26	27	34	35
47	56	57	67

(a). Initial state matrix

30	7C	77	7B
0C	13	EC	5F
7D	FA	59	47
F7	CC	18	96
A0	B1	5B	85

(b). After byte substitution

77	7B	30	7C
5F	0C	13	EC
FA	59	47	7D
96	F7	CC	18
A0	B1	5B	85

(c). After row diffusion

FIGURE 9. (a). Initial state matrix. (b). After byte substitution. (c). After row diffusion.

77	78	30	76
51	06	13	66
14	59	47	74
96	17	66	18
40	81	58	85

(a). After TRIFID cipher

77	51	14	96	40
78	06	59	17	81
30	13	47	66	58
76	66	74	18	85

(b). After transposition

20	DD	F0	70	C5
A1	0E	1B	0A	34
F5	DB	5F	65	E5
3D	2A	CA	E0	08

(c). After mix-column diffusion

FIGURE 10. (a). After TRIFID cipher. (b). After transposition. (c). After mix-column diffusion.

a) Performing TRIFID cipher on 'A':
Let's assume key = V $ Q A W S E D R F T G Y H U J I K O L P Z M X N C B

Square matrix-1			Square matrix-2			Square matrix-3		
V	$	Q	F	T	G	O	L	P
A	W	S	Y	H	U	Z	M	X
E	D	R	J	I	K	*N*	C	B

'A' corresponds to the state "xyz" = 211

b) Performing TRIFID cipher on 'B':
Let's assume key = Q A W S E D R F T G Y H U J I K $ O L P Z M X N C B V

Square matrix-1			Square matrix-2			Square matrix-3		
Q	A	W	G	Y	H	L	P	Z
S	E	D	U	J	I	M	X	N
R	F	T	K	$	O	C	**B**	N

'B' corresponds to the state "xyz"= 323

c) Performing TRIFID cipher on 'C':
Let's assume key= O L P Z M X N C B V $ Q A W S E D R F T G Y H U J I K

Square matrix-1			Square matrix-2			Square matrix-3		
O	L	P	V	$	Q	F	T	G
Z	M	X	A	W	S	Y	H	U
N	**C**	B	E	D	R	J	I	K

'C' corresponds to the state "xyz"=321

d) Performing TRIFID cipher on 'D':
Let's assume key = G Y H U J I K $ O L P Z M X N C B V Q A W S E D R F T

Square matrix-1			Square matrix-2			Square matrix-3		
G	Y	H	L	P	Z	Q	A	W
U	J	I	M	X	N	S	E	**D**
K	$	O	N	B	V	R	F	T

'D' corresponds to the state "xyz"=233

e) Performing TRIFID cipher on 'E':
Let's assume key = F T G Y H U J I K O L P Z M X N C B V $ Q A W S E D R

Square matrix-1			Square matrix-2			Square matrix-3		
F	T	G	O	L	P	V	$	Q
Y	H	U	Z	M	X	A	w	S
J	I	K	N	C	B	**E**	D	R

'E' corresponds to the state "xyz"= 313

f) Performing TRIFID cipher on 'F':
Let's assume key = L P Z M X N C B V Q A W S E D R F T G Y H U J I K $ O
'F' corresponds to the state "xyz" = 322

Square matrix-1			Square matrix-2			Square matrix-3		
L	P	Z	Q	A	W	G	Y	H
M	X	N	S	E	D	U	J	I
C	B	V	R	**F**	T	K	$	O

- Equivalent digit corresponding to each unique alphabet is computed based on stego-key4. Let's assume the stego-key4 is "001 000 010 101 001 010". Here, groups of three bits from left to right represent the key-bits for alphabets 'A' to' F' respectively. The group of 3 bits decides the mode (shown in Fig. 5) for computing equivalent digit for an alphabet. Here for each alphabet, the combination of 3 key-bits is chosen such that the computation of the corresponding state "xyz" should not exceed 15 in decimal. The computation of equivalent digit for each unique alphabet is performed based on the modes shown in Fig.5. Table 3 shows the corresponding key bits for all unique alphabets of the matrix and their computed equivalent digits.
- The state matrix is updated based on the output of the previous step. The updated matrix is shown in Fig. 10(a). Further, the updated matrix is transposed as shown in Fig. 10(b).
- For each column of transposed matrix, the mix column diffusion is performed by using a circulant MDS (Maximum Distance Separable) matrix (as used in AES).

TABLE 3. Digits equivalent of alphabets computed based on stego-key4.

Alphabets	A	B	C	D	E	F
Corresponding state (output of TRIFID cipher)	211	323	321	233	313	322
Key bits (stego-key4)	001	001	000	010	101	010
Computed digit	4	8	6	4	6	1

The mix-column operation for the first column of the transposed matrix is as follows:

$$\begin{bmatrix} Eo \\ E1 \\ E2 \\ E3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} 77 \\ 78 \\ 30 \\ 76 \end{bmatrix} = \begin{bmatrix} 20 \\ A1 \\ F5 \\ 3D \end{bmatrix}$$

Similarly, mix-column operations for other columns of transposed matrix are also performed. Post mix-column operation, the modified matrix is shown in Fig. 10(c).

- All elements of each column are concatenated based on steo-key5. Since, there are five columns in the matrix, therefore the size of stego-key5 is of 15 bits. For ach column, the combination of 3 bits decides the mode for concatenation of corresponding elements (shown in Fig. 5). Let's assume the stego-key5 is "001 000 010 101 000". Here, groups of three bits from left to right represent the key-bits for column-1 to column-5 respectively. For this key, the final sequence of elements is as follows:
  E0E1E3E2E0E1E2E3E0E2E1E3E0E3E2E E0E1E2E3
  20A13DF5DD0EDB2AF05F1BCA70E0650AC534E508
- The corresponding bit-stream is: 0010-0000-1010-0001-0011-1101-1111-0101-1101-1101-0000-1110-1101-1011-0010-1010-1111-0000-0101-1111-0001-1011-1100-1010-0111-0000-1110-0000-0110-0101-0000-1010-1100-0101-0011-0100-1110-0101-0000-1000.
- Let's assume the designer selected bit-stream length is 48. Hence, the truncated bit-stream is as follows (the bit-stream has total 24 number of 0s and 24 number of 1s): "001000001010000100111101111101011101110100001110"
- Based on the encoding meaning (shown in table 1) of '0' and '1' bits of the bit-stream, the potential stego-constraints to be implanted in to the design are obtained. The stego-constraints corresponding to 24 number of 0s in the selected size of bit-stream are as follows:
  <V0, V2>, <V0, V4>, <V0, V6>, <V0, V8>, <V0, V10>, <V0, V12>, <V0, V14>, <V0, V16>, <V0, V18>, <V0, V20>, <V0, V22>, <V2, V4>, <V2, V6>, <V2, V8>, <V2, V10>, <V2, V12>, <V2, V14>, <V2, V16>, <V2, V18>, <V2, V20>, <V2, V22>, <V4, V6>, <V4, V8>, <V4, V10>

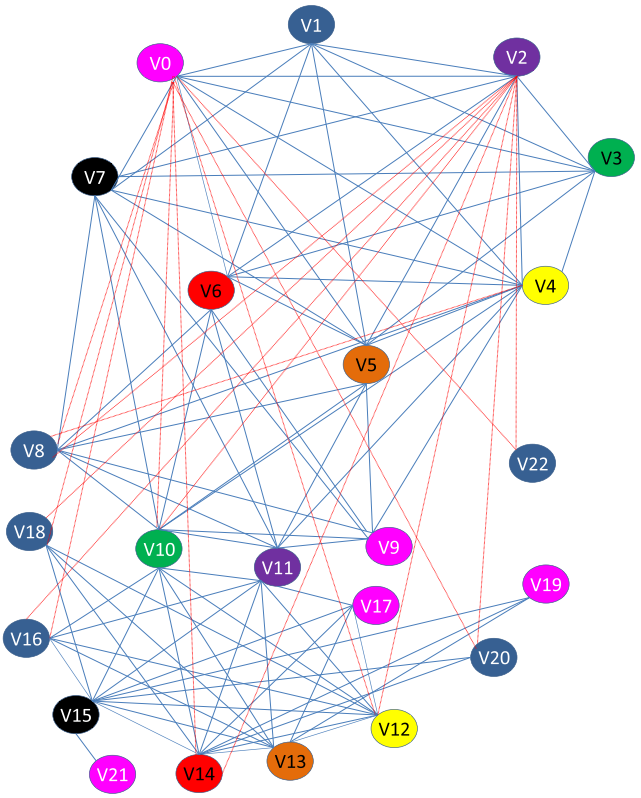


FIGURE 11. The CIG of 8-point DCT after implanting steganography during register allocation phase Note: Dotted red lines show the stego-constraints implanted (corresponding to 0s).

Further, out of total 24 number of 1s in the truncated bit-stream, embedding of only 15 number of 1s is possible. This is because, according to the encoding rule of bit '1' (shown in table 1), operations are allocated to the functional unit of specific vendor type to embed stego-constraints corresponding to bit '1'. In this case, total operations are 15, therefore embedding of maximum 15 umber of 1's is possible.

- Stego-constraints corresponding to bit '0' are implanted in the register allocation phase of HLS and stego-constraints corresponding to bit '1' are implanted in the hardware (functional unit) allocation phase of HLS. Post embedding phase-1 steganography (stego-constraints corresponding to '0'), the CIG and register allocation table are shown in Fig. 11 and Table 4 respectively (highlighted register columns indicate change due to implanted steganography). Once, phase-1 steganography is embedded, phase-2 steganography is performed by embedding stego-constraints corresponding to bit '1'. Out of 24 number of 1s, embedding of only 15 number of 1s is possible, as explained earlier. However, out of 15 number of 1s, only 12 times 1s are effectively embedded. This is because, stego-constraints corresponding to three 1s do not satisfy (FU reallocation is not possible for operation 10, 12 and 14 because these operations should be assigned to FU of even vendor type (according to the

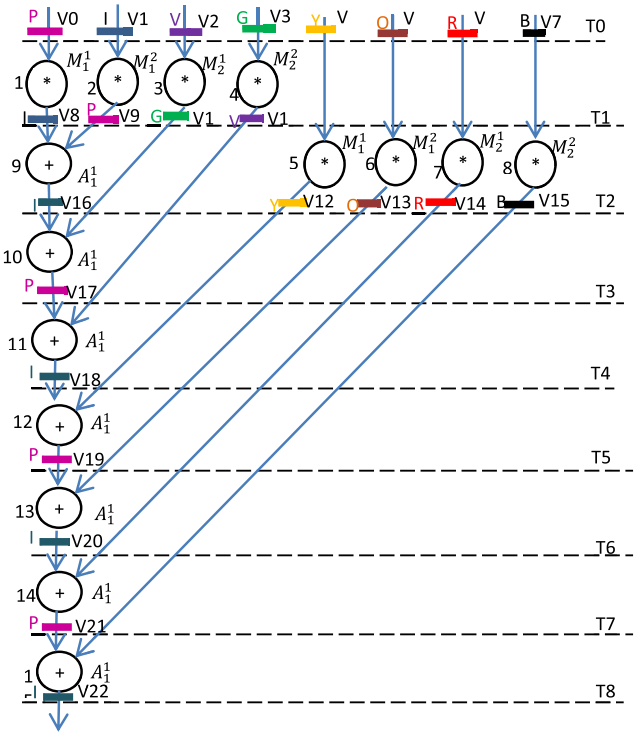


FIGURE 12. Scheduled and hardware allocated 8-point DCT using 1(+) and 4(*) after implanting steganography during both pase-1 and phase-2.

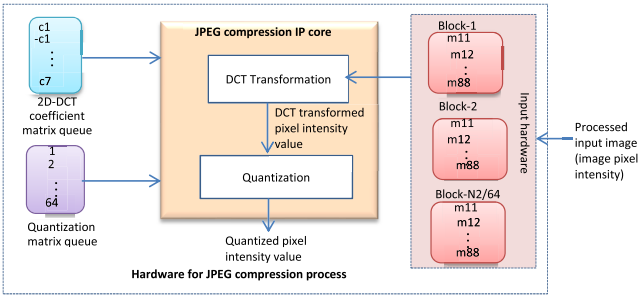


FIGURE 13. Hardware of JPEG compression process.

encoding rule of bit '1'), however this assignment is not possible as there is only 1 adder of odd vendor type. Post embedding dual phase steganography, the final scheduled DFG of DCT core is shown in Fig. 12. The observation of embedding dual phase steganography for stego-constraint size of 48 bits is shown in Table 5.

- Total size of stego-key for JPEG DCT core is computed to be "**610 bits**" from (1).

IV. DESIGNING JPEG CODEC WITH EMBEDDED HARDWARE STEGANOGRAPHY

A. DEMONSTRATION OF EMBEDDING HARDWARE STEGANOGRAPHY ON DCT CORE USED IN JPEG COMPRESSION

In JPEG CODEC process, an image to be compressed is first converted into an N × N matrix, where each entry in the matrix represents the pixel intensity at corresponding location in the image. The pixel intensity ranges from 0 to 255. Since,

$$\begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & -C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix}$$

FIGURE 14. 2D- DCT coefficient matrix.

TABLE 4. Register/color allocation of storage variables of 8-point DCT after implanting steganography.

T	Pink	Indigo	Violet	Green	Yellow	Orange	Red	Black
0	V0	V1	V2	V3	V4	V5	V6	V7
1	V9	V8	V11	V10	V4	V5	V6	V7
2	--	V16	V11	V10	V12	V13	V14	V15
3	V17	--	V11	--	V12	V13	V14	V15
4	--	V18	--	--	V12	V13	V14	V15
5	V19	--	--	--	--	V13	V14	V15
6	--	V20	--	--	--	--	V14	V15
7	V21	--	--	--	--	--	--	V15
8	--	V22	--	--	--	--	--	--

the DCT core underneath JPEG CODEC processor processes an 8 × 8 matrix at a time, therefore the NxN matrix representing input image is divided in to a number of non-overlapping 8 × 8 blocks of pixels. Further, since DCT can process the pixel value within the range of −128 to +127, therefore each pixel value in all 8 × 8 blocks of pixels is leveled off by subtracting 128. Block diagram of the hardware of JPEG compression process is shown in Fig. 13. As shown in the figure, DCT transformation is performed on each 8 × 8 blocks of pixels using a 2D-DCT coefficient matrix. The generic 2D-DCT coefficient matrix is shown in Fig. 14. Post-DCT transformation of each 8 × 8 blocks of pixels, quantization (at Q90 for higher quality) is performed for compression. By choosing specific quantization matrix, different ratio of compression and quality can be obtained. The quality level ranges from 1 to 100, where 1 indicates highest compression but poorest quality and 100 indicates lowest compression with best quality. In the proposed approach, the quantization matrix Q90 (quality level 90) is used to compress the CT scan medical images within the acceptable compression ratio. Once a DCT transformed matrix is quantized using quantization matrix, zigzag scanning is performed to obtain the data into one-dimensional array. Thereafter, the compressed data is converted into a bit stream by applying run-length encoding for storing and transmitting. In the process of decompression/reconstruction of original image, the stored image data (in compressed form) is processed through run-length decoding followed by inverse zigzag scanning. This results into a quantized version of image data in the form of matrix. Next, inverse quantization is performed using corresponding quantization matrix which gives de-quantized data of pixels intensities. Thereafter, inverse DCT is performed on de-quantized data which gives the final pixel intensity values of reconstructed image.

TABLE 5. Observations of embedding stego-constraints into the DCT core.

Vendor's bit-string representing stego-constraints	Additional edges to be inserted between nodes in the CIG (Phase-1)	Allocate FU type (Phase-2)	Observations
0	<V0, V2>	--	Edge exists by default
0	<V0, V4>	--	Edge exists by default
1	--	opn 1→U1	FU allocated by default
0	<V0, V6>	--	Edge exists by default
0	<V0, V8>	--	New edge to be added
0	<V0, V10>	--	New edge to be added
0	<V0, V12>	--	New edge to be added
0	<V0, V14>	--	New edge to be added
1	--	opn 2→U2	FU reallocation to be done
0	<V0, V16>	--	New edge to be added
1	--	opn 3→U1	FU reallocation to be done
0	<V0, V18>	--	New edge to be added
0	<V0, V20>	--	New edge to be added
0	<V0, V22>	--	New edge to be added
0	<V2, V4>	--	Edge exists by default
1	--	opn 4→U2	FU allocated by default
0	<V2, V6>	--	Edge exists by default
0	<V2, V8>	--	New edge to be added
1	--	opn 5→U1	FU allocated by default
1	--	opn 6→U2	FU reallocation to be done
1	--	opn 7→U1	FU reallocation to be done
1	--	opn 8→U2	FU allocated by default
0	<V2, V10>	--	New edge to be added
1	--	opn 9→U1	FU allocated by default
1	--	opn 10→U2	FU reallocation not possible
1	--	Opn11→U1	FU allocated by default
1	--	opn 12→U2	FU reallocation not possible
1	--	opn 13→U1	FU allocated by default
0	<V2, V12>	--	New edge to be added
1	--	opn 14→U2	FU reallocation not possible
0	<V2, V14>	--	New edge to be added
1	--	opn 15→U1	FU allocated by default
1	--	NA	NA
1	--	NA	NA
0	<V2, V16>	--	New edge to be added
1	--	NA	NA
1	--	NA	NA
1	--	NA	NA
0	<V2, V18>	--	New edge to be added
1	--	NA	NA
0	<V2, V20>	--	New edge to be added
0	<V2, V22>	--	New edge to be added
0	<V4, V6>	--	Edge exists by default
0	<V4, V8>	--	New edge to be added
1	--	NA	NA
1	--	NA	NA
1	--	NA	NA
0	<V4, V10>	--	Edge exists by default

The DCT transformation of each 8×8 block of pixels is performed by following matrix multiplication.

$$X = (T^{*}M)^{*}T' \tag{2}$$

where T represents the 2D-DCT coefficient matrix (shown in Fig. 12), M represents the 8×8 block of pixels of input image and T' represents the transpose of T. The first pixel value 'X11' of the DCT transformed matrix X is given as follows:

$$X11 = c4 * d11 + c4 * d12 + c4 * d13 + c4 * d14 + c4 * d15 + c4 * d16 + c4 * d17 + c4 * d18 \tag{3}$$

where, eight times c4 value in the equation represents the first column of matrix T' and d11, d12, ...d18 represent the elements of first row of the matrix $T^{*}M$. The first element d11 is calculated as follows:

$$d11 = c4 * m11 + c4 * m21 + c4 * m31 + c4 * m41 + c4 * m51 + c4 * m61 + c4 * m71 + c4 * m81 \tag{4}$$

where, eight times c4 value in the equation represents the first row of matrix T and m11, m21, ...m81 represent the elements of first column of the input matrix M. Similarly, other elements of $T^{*}M$ matrix are calculated. Thus obtained $T^{*}M$ matrix is used to evaluate the matrix 'X' using (2). Computing hardware of matrix $T^{*}M$ is represented as micro IP which is further used in macro IP representing computing hardware of matrix X. Figure 15 shows the DFG of JPEG image compression as macro IP which uses micro IPs underneath. As shown in Fig.15, the first pixel value after DCT transformation is generated after operation 135. The first pixel of compressed JPEG image is generated by operation 136 which quantizes the DCT transformed value (output of operation 135) using corresponding element q1 of quantization matrix Q90 (as shown in Fig. 15).

Note: Decompression is just the inverse of compression process. Therefore, a similar hardware design with different inputs can be used. For the sake of brevity, details of decompression process have not been included.

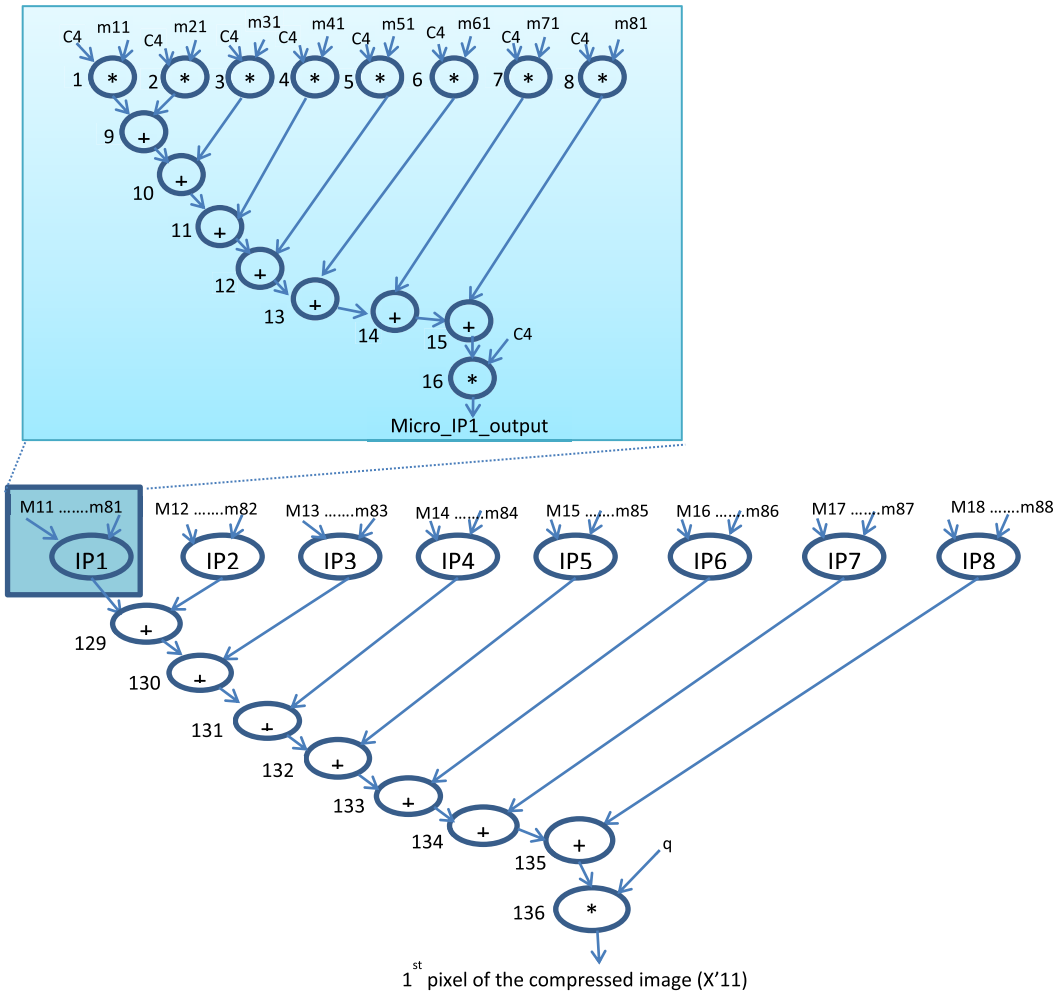


FIGURE 15. DFG of un-obfuscated JPEG compression IP core as macro IP comprising 8 micro-IPs underneath.

B. STRUCTURAL OBFUSCATION ON JPEG CODEC- 1ST LINE OF DEFENSE

As shown in Fig. 15, the un-obfuscated DFG of JPEG image compression has been presented as a macro IP which comprises of 8 micro IPs underneath [13]. Here structural obfuscation is performed at two levels i.e. both micro IP and macro IP are structurally obfuscated. After performing tree height transformation (THT) based structural obfuscation, the obfuscated JPEG compression DFG is shown in Fig. 16. In THT based structural obfuscation, some sequentially executing operations are forced to execute as parallel sub-computations, thus incorporating the structural obfuscation by altering the data dependency of operations. For example, data dependencies of six operations (opn 10, 11, 12, 13, 14 and 15) in micro IP1 get altered owing to THT based structural obfuscation as shown in Fig. 16. Similarly, data dependencies of six operations in each other micro IPs also get altered. When THT based structural obfuscation is performed on macro IP, data dependencies of operations 130, 131, 132, 133, 134 and 135 get altered. This would result into structural obfuscation in the form of change in interconnectivity of resources, multiplexer and de-multiplexer inputs/outputs in datapath [13]. Thus, datapath architecture and controller logic of the design would be obscured without change in functionality. The structurally obfuscated design becomes unobvious for an attacker to interpret. Thus, the structure of the design remains concealed for the attacker and he/she fails to insert malicious logic (Trojan) and counterfeit/clone the hardware. Thereby, structural obfuscation acts as 1st line of defense.

C. INTEGRATING HARDWARE STEGANOGRAPHY WITH STRUCTURAL OBFUSCATION FOR JPEG COMPRESSION PROCESSOR- 2ND LINE OF DEFENSE

The proposed crypto-based dual-phase steganography is integrated to the structurally obfuscated JPEG CODEC as 2nd line of defense. This enhances the security of JPEG compression processor against aforementioned hardware threats. The process of embedding crypto-based dual-phase steganography for JPEG compression is as follows:

1. The constraints based list scheduling and hardware allocation are performed on structurally obfuscated DFG of JPEG compression. Scheduling is performed based on resource constraints (3+, 3*), where maximum two instances of a

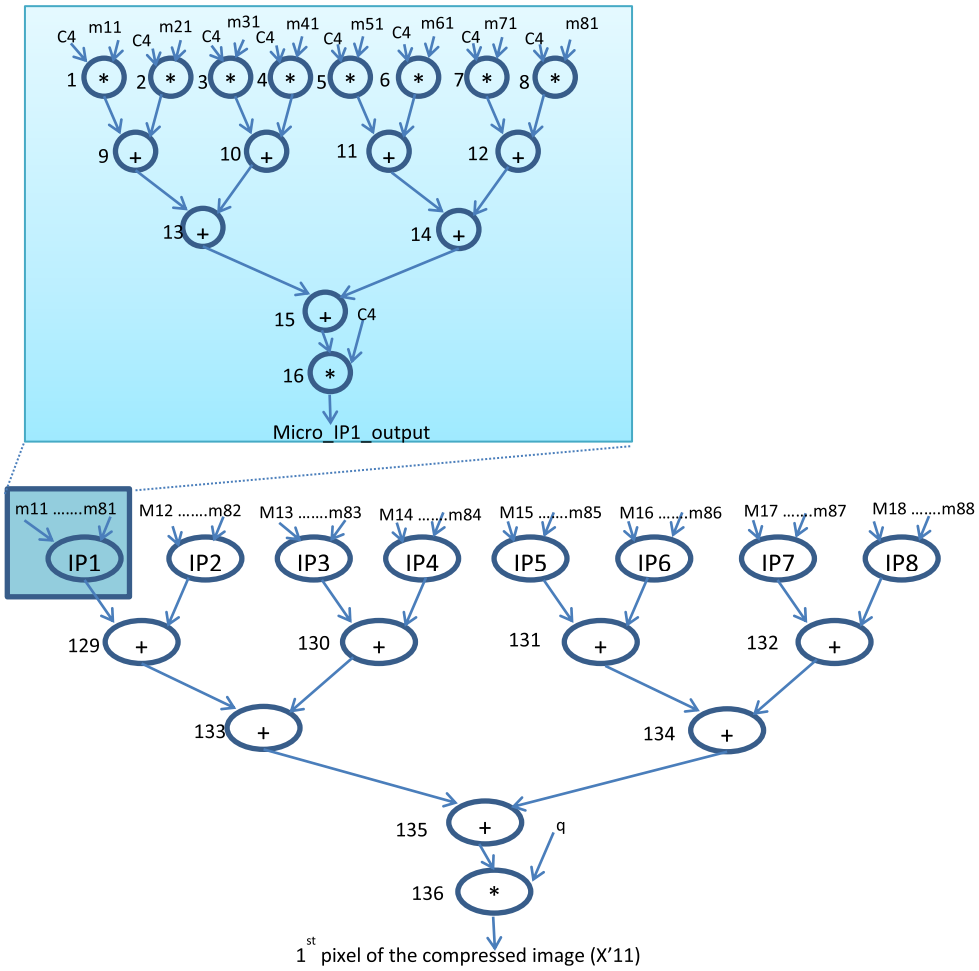


FIGURE 16. Structurally obfuscated DFG of JPEG CODEC IP core.

functional unit is of vendor type-1 and one instance is of vendor type-2.

2. Register allocation/CIG is obtained from scheduled obfuscated DFG. Both register allocation table and scheduled obfuscated DFG act as cover design data for embedding proposed steganography.

3. The secret design data for steganography is obtained from the register allocation of the obfuscated JPEG design (process of obtaining the secret design data is same as explained for DCT core in section 3.3 earlier).

4. The cover design data, secret design data and stego-keys are fed to the stego-encoder system. The secret design data and stego-keys are used to generate stego-constraints to be embedded into the cover design data. Suppose, designer selected stego-keys are as follows:

- Stego-key1 = 001 (key size = 3bits)
- Stego-key2 = 11-10-00-01-00-10-10-10-11-10-00-00-10-01-11-11-11-11-10-00-00-10-10-11-01-11-11-01-11-01-00-11-11-11-00-11-01-11 (key size = 2*38 bits)
- Stego-key3 =
  For alphabet 'a' = v$qawsedrftgyhujikolpzmxncb
  For alphabet 'b' = qawsedrftgyhujik$olpzmxncbv
  For alphabet 'c' = olpzmxncbv$qawsedrftgyhujik
  For alphabet 'd'= gyhujik$olpzmxncbvqawsedrft
  For alphabet 'e'= ftgyhujikolpzmxncbv$qawsedr
  For alphabet 'f'= lpzmxncbvqawsedrftgyhujik$o
  (key size = 6*(⌈ ($\log_2(27!)$) ⌉ bits)
- Stego-key4 = 010-001-100-101-011-001
  (key size = 6*3 bits)
- Stego-key5 = 000-001-010-011-100-101-001-011-010-100-100-000-100-100-011-010-001-000-100-101-011-010-001-000-101-011-001-000-100-101-011-010-001-011-101-011-011-100
  (key size = 38*3 bits)
- **Total size of the key** (from eq. (1)) = 3+(2*38) + 6*(⌈ ($\log_2(27!)$)⌉+ (6*3) + (38*3) = 3+76+564+18+114 = 775 bits

5. Based on the stego-keys, the final bit-stream is generated using the same steps as described in section 3.2.

6. Let's assume the designer selected stego-constraints size is 400. Therefore, bit-stream of size 400 bits is truncated from the final bit-stream obtained in previous step. For this size, bit '0' appears total 197 times and bit '1' appears total 203 times in the truncated bit stream. Stego-constraints corresponding

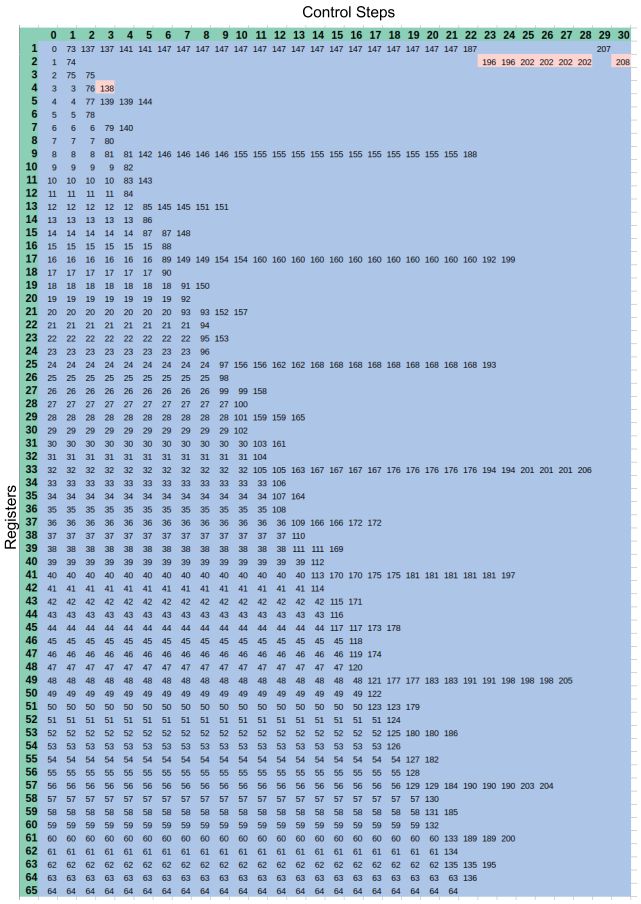


FIGURE 17. Register allocation of storage variables (0 to 208) of JPEG compression hardware after steganography Note: Total register count is 73, however partial register allocation till 65 registers is shown (for sake of brevity).

to bit '0' are embedded into the register allocation phase. Post embedding bit '0', the modified register allocation of JPEG compression is shown in Fig. 17. Further, stego-constraints corresponding to bit '1' are embedded into the hardware (functional unit) allocation phase. According to the encoding rule of bit '1' (shown in table 1), the odd operations are allocated to functional unit of odd vendor type and even operations are allocated to functional unit of even vendor types. This reallocation of functional unit reflects the embedded stego-constraints corresponding to bit '1'. However, out of 203, some of bit '1' may not be embedded. For example, there are total 136 operations in case of JPEG design (as shown in Fig. 16), therefore embedding of maximum 136 number of 1s is possible. Further, out of 136, only 111 times 1s are effectively embedded. This is because, stego-constraints corresponding to some number of 1s do not satisfy (FU reallocation is not possible for some operations as these cannot be assigned to FU of vendor type according to encoding rule of bit '1'). Therefore only 111 number of 1s are effective embedded. Post embedding 1s into the hardware allocation phase, scheduling and FU allocation of steganography embedded obfuscated JPEG compression is shown in Table 6. Note: For the sake of brevity, the pre-steganography JPEG compression design has not been included. Post embedding constraints into two distinct phases of HLS, the steganography embedded obfuscated JPEG processor is generated which is highly secured using double line of defense.

TABLE 6. Scheduling of JPEG compression hardware after implanting steganography.

T	Operations assign to M_1^1	Operations assign to M_2^1	Operations assign to M_1^2	Operations assign to A_1^1	Operations assign to A_2^1	Operations assign to A_1^2
1	1	3	2	--	--	--
2	5	6	4	9	--	--
3	7	17	8	11	--	10
4	19	20	18	13	--	12
5	21	23	22	25	26	14
6	33	34	24	15	27	29
7	35	37	36	41	--	28
8	39	40	38	42	--	30
9	49	51	50	43	45	44
10	53	54	52	31	57	46
11	55	65	56	59	47	58
12	67	68	66	61	--	60
13	69	71	70	73	74	62
14	81	82	72	63	75	77
15	83	85	84	89	--	76
16	87	88	86	90	--	78
17	97	99	98	91	93	92
18	101	102	100	79	105	94
19	103	113	104	95	107	106
20	115	116	114	--	109	108
21	117	119	118	121	122	110
22	32	120	16	111	123	125
23	64	80	48	129	--	124
24	112	--	96	130	--	126
25	--	--	--	127	131	133
26	--	--	128	--	--	--
27	--	--	--	--	--	132
28	--	--	--	--	--	134
29	--	--	--	135	--	--
30	--	136	--	--	--	--

D. DEMONSTRATION OF END TO END SECURED JPEG CODEC PROCESS

Fig. 18 shows the generic block diagram of end to end secured JPEG CODEC process that ensures the security of medical images. As shown in the figure, the proposed steganography embedded obfuscated JPEG compression processor generates secured compressed image data quantized at quality level of Q90. To reconstruct the image, the compressed/quantized image data is decompressed/de-quantized using proposed steganography embedded obfuscated JPEG de-compression processor. Thereby the proposed JPEG CODEC processor secures the medical images from being corrupted/altered by external hardware threats and ensures the reconstruction of medical images in their genuine form. The compression and de-compression on six medical images of CT scan dataset [19] are performed for demonstration. The original, quantized (at quality level of Q90) and reconstructed CT scan images are shown in Fig. 19. Table 7 shows the Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) of compressed images for quantization level of Q90 to ensure acceptable quality.

V. EXPERIMENTAL RESULTS

The proposed approach has been implemented in C++ and run on processor with 4 GB, DDR3 memory at 1.9 GHz. This

FIGURE 18. End to end JPEG CODEC process.

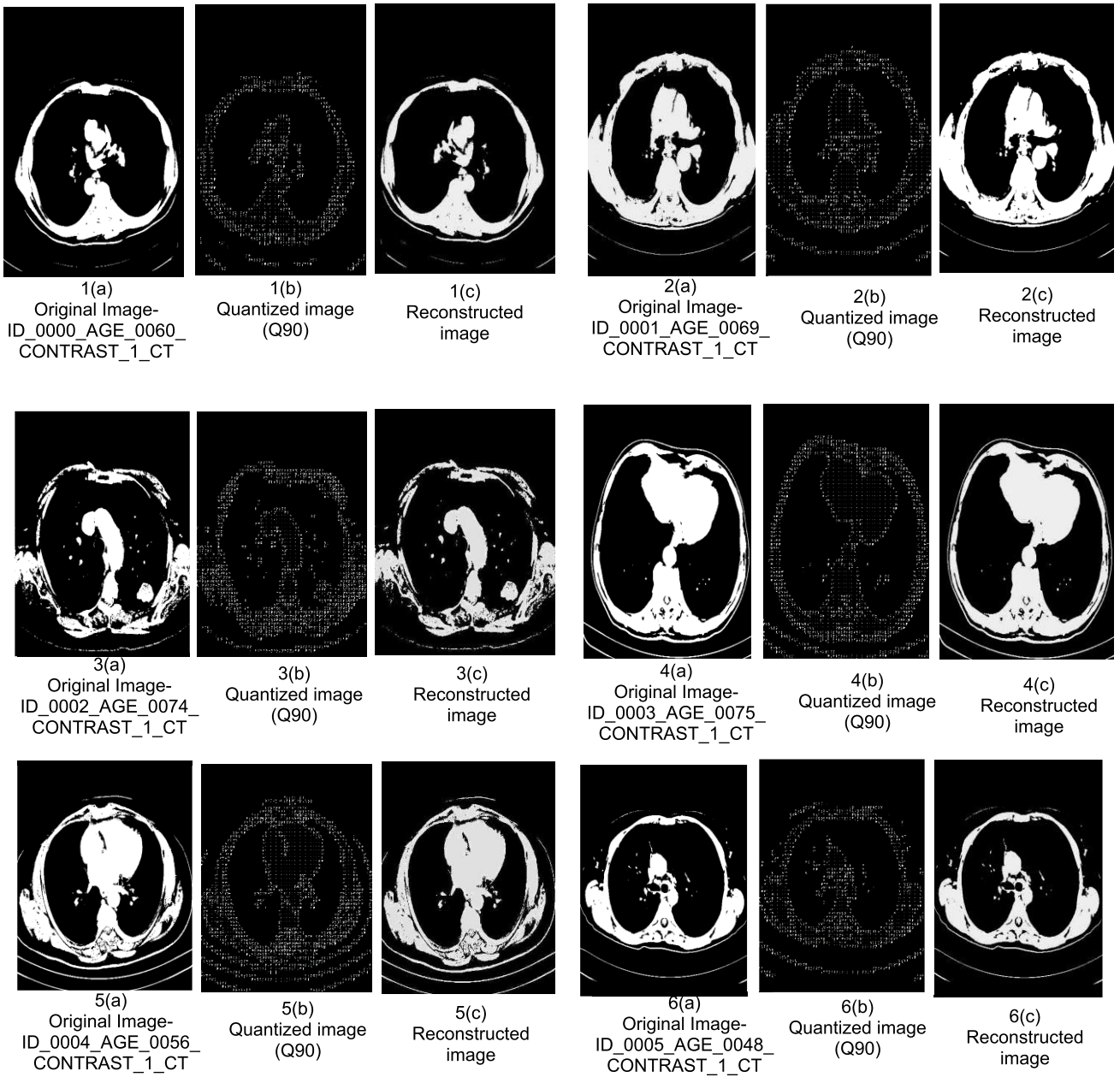


FIGURE 19. Original, quantized (at quality level Q90) and reconstructed CT scan images using JPEG CODEC hardware.

section analyses the security proposed approach obtained from structural obfuscation (1st line of defense) and hardware steganography (2nd line of defense). The security due to structural obfuscation is analyzed in terms of strength of obfuscation. Further, the security due to steganography is analyzed using probability of coincidence metric for different design solutions (resource constraints) and key-size of JPEG compression. In addition, for each design solution, the security is evaluated for varying size of designer selected stego-constraints. The proposed approach achieves higher

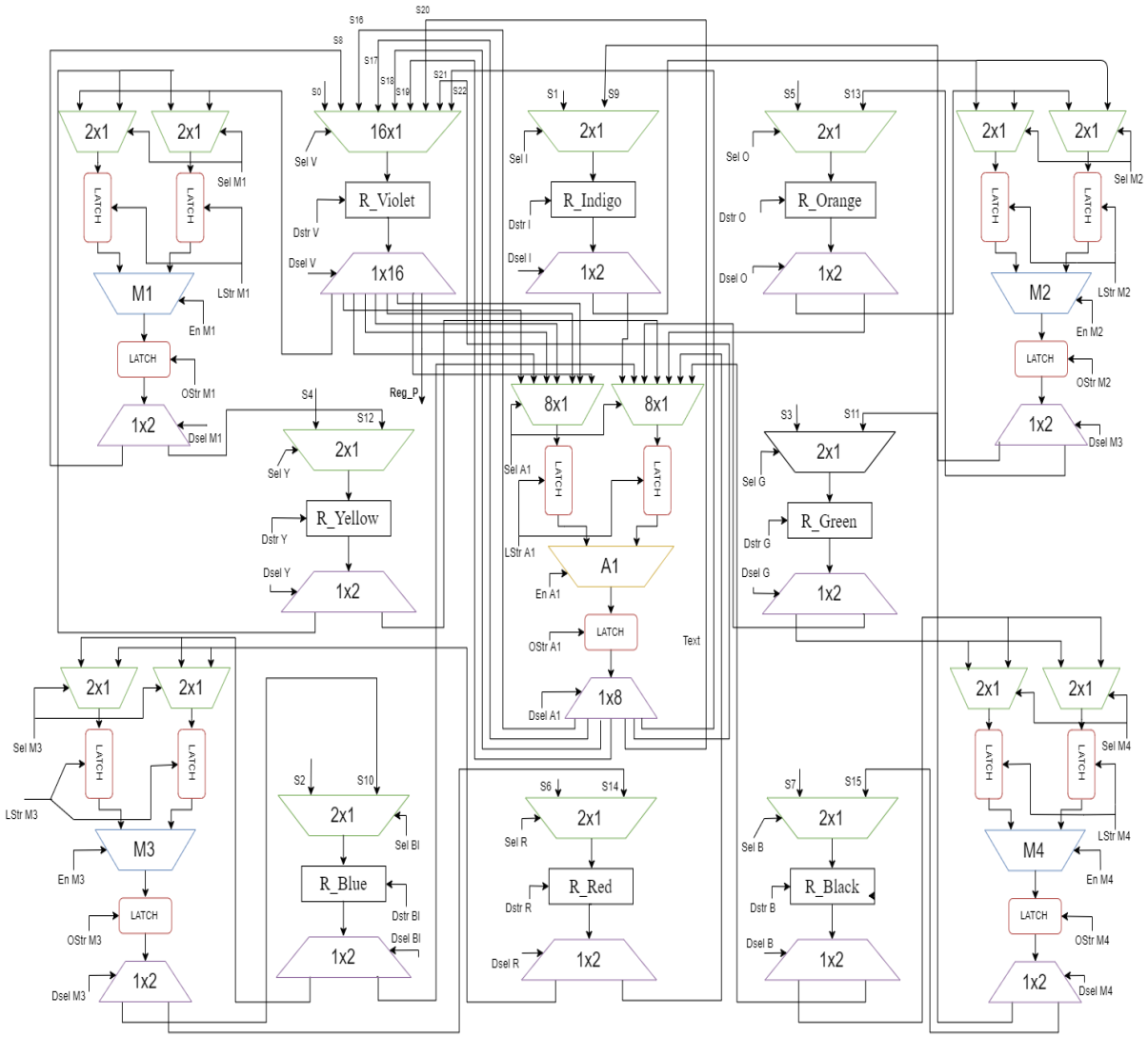


FIGURE 20. Un-obfuscated JPEG DCT core.

TABLE 7. Results of compressed CT scan images [19] using proposed secured JPEG CODEC.

Original Image Name	PSNR	MSE
ID_0000_AGE_0060_CONTRAST_1_CT	22.5965	1.2375
ID_0001_AGE_0069_CONTRAST_1_CT	21.5139	1.5878
ID_0002_AGE_0074_CONTRAST_1_CT	20.0345	2.2322
ID_0003_AGE_0075_CONTRAST_1_CT	21.1257	1.7362
ID_0004_AGE_0056_CONTRAST_1_CT	19.1805	2.7173
ID_0005_AGE_0048_CONTRAST_1_CT	22.1125	1.3834

security of JPEG processor on very low design cost. The design cost is also assessed for different design solutions and varying size of stego-constraints.

A. SECURITY ANALYSIS OF THE PROPOSED SECURE JPEG COMPRESSION PROCESSOR

1) STRENGTH OF STRUCTURAL OBFUSCATION

The structural obfuscation results into **changes in functional units such as adders, multipliers etc., storage elements (register count, latches etc.) and interconnectivity of resources, multiplexer and de-multiplexer inputs/outputs, without affecting the functionality**. This obscures the datapath architecture and controller logic of the design without change in functionality from an attacker's perspective. Thus, structurally obfuscated JPEG design becomes unobvious for an attacker to interpret. For example, for JPEG DCT core, the un-obfuscated and obfuscated versions are shown in Fig. 20 and Fig. 21 respectively. Un-obfuscated version indicates JPEG processor design which does not have any security algorithm employed. In other words, it is an unsecured version that does not have capability to counter RE (resulting into Trojan insertion). Hence, such a JPEG processor is not tamper resistant and trustworthy for usage in medical imaging systems. On the other hand, obfuscated version is tamper resistive to RE (resulting into Trojan insertion), hence trustworthy for usage in medical imaging systems. It is evident from Fig. 20 and Fig. 21 that the structural

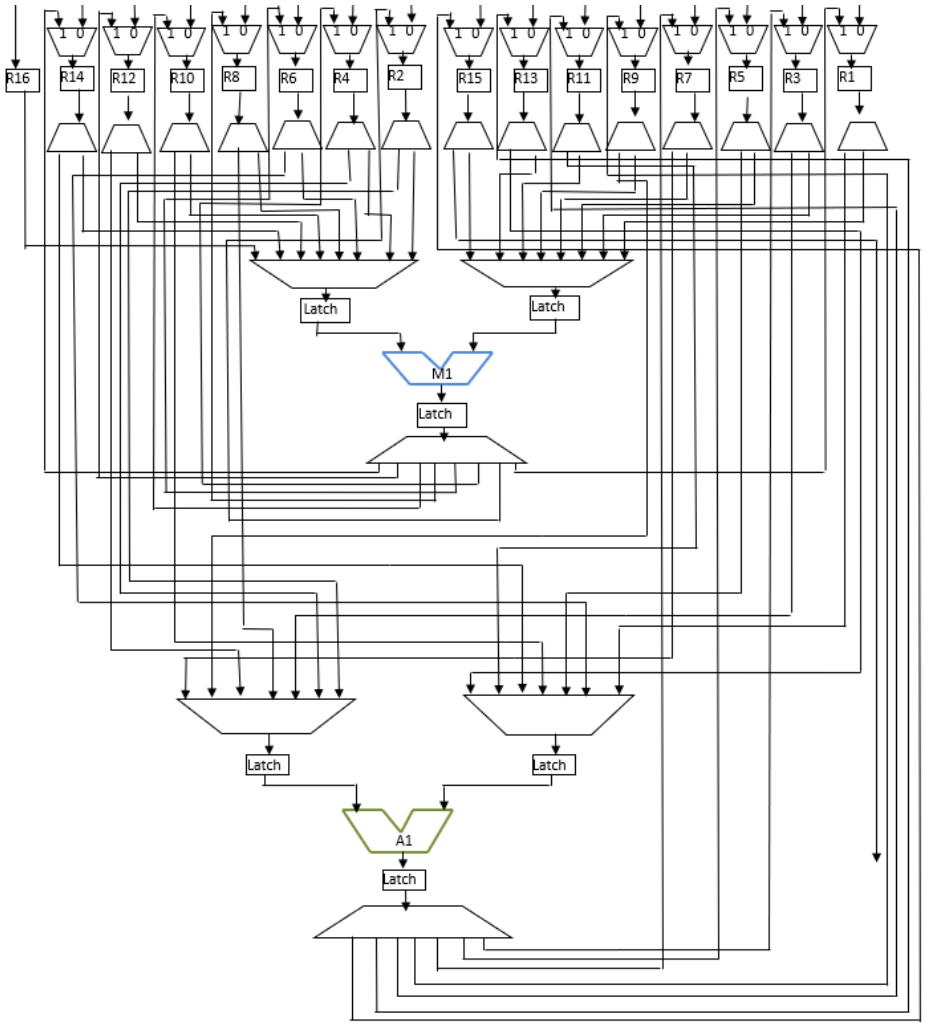


FIGURE 21. Obfuscated JPEG DCT core.

obfuscation introduces significant changes in the structure of the design, thus making it harder to reverse engineer. As explained earlier, the obfuscated JPEG DCT core achieves significant structural obfuscation through proposed technique resulting into changes in functional units such as adders, multipliers etc., storage elements (register count, latches etc.) and interconnectivity of resources, multiplexer and de-multiplexer inputs/outputs. As the structure of the design remains concealed for the attacker, therefore he/she fails to insert malicious logic (Trojan) and counterfeit/clone the hardware. Table 8 compares the JPEG CODEC design pre and post structural obfuscation for resource constraints (3+, 3*). As shown in the table, respective gates affected at gate-level design is 10064 without changing the functionality.

2) STRONGER DIGITAL EVIDENCE HIDDEN IN STEGO-IMPLANTED JPEG COMPRESSION PROCESSOR

It signifies the strength of the implanted steganography information in the obfuscated JPEG compression processor. More steganography information distributed uniformly across the design in a covert manner implies that the design carries stronger digital evidence for authentication purpose. Stronger digital evidence (more steganography) hidden inside the design secures an authentic JPEG processor against possible removal and tampering. Thus, stronger steganography enhances the possibility of detecting counterfeited/cloned/tampered JPEG compression ICs. The proposed approach embeds crypto-based steganography at two distinct phases of HLS, therefore generates strong hidden digital evidence for JPEG compression.

3) LOWER PROBABILITY OF COINCIDENCE (PC) IN STEGO-IMPLANTED JPEG COMPRESSION PROCESSOR

The security due to embedding proposed hardware steganography into the obfuscated JPEG compression processor is assessed using probability of coincidence (Pc) metric [16], [17]. The Pc metric indicates the probability of obtaining the proposed steganography information in a non-steganography design. Since in the proposed approach, the obtained Pc value will be extremely low, thus this indicates that the chance

TABLE 8. Comparison of JPEG hardware pre and post structural obfuscation.

	Resource configuration	Structural changes due to proposed obfuscation
Non-obfuscated JPEG hardware	4+, 8*, 12(8:1) mux, 12(16:1) mux, 6 (1:8 demux), 6 (1:16 demux)	10064 gates
Structurally obfuscated JPEG hardware	3+, 3*, 10(32:1) mux, 2(16:1) mux, 5 (1:32 demux), 1 (1:16 demux)	

TABLE 9. Effective number of 0's and 1's for different size of stego-constraints.

Design solution (resource constraint)	Total # of constraint =100		Total # of constraint =200		Total # of constraint =300		Total # of constraint =400	
	Effective # of 0's embedded	Effective # of 1's embedded	Effective # of 0's embedded	Effective # of 1's embedded	Effective # of 0's embedded	Effective # of 1's embedded	Effective # of 0's embedded	Effective # of 1's embedded
3+, 3*	42	49	89	93	139	111	197	111
3+, 5*	48	48	97	94	148	122	203	122
5+, 5*	49	49	98	93	153	124	208	124
7+, 9*	40	59	87	108	135	131	186	131
9+, 9*	44	55	93	104	142	132	188	132
11+, 11*	53	46	109	89	160	131	217	131

of carrying the proposed steganography information in a cloned/counterfeited design coincidentally is almost negligible. Thus, cloned/counterfeited JPEG design can be easily detected as these versions would not carry the authentic stego-information of the proposed approach.

The impact on security of JPEG compression due to both phase-1 and phase-2 steganography has been analyzed using Pc metric. Post phase-1 steganography, the probability of coincidence metric is evaluated as follows [16], [17]:

$$Pc^1 = \left(1 - \frac{1}{c}\right)^{f1} \tag{5}$$

where, Pc^1 represents the probability of coincidence post phase-1 steganography, c denotes the number of colors/registers in the JPEG compression design before embedding steganography and f1 denotes the number of effective stego-constraints (effective number of 0s) embedded in the CIG/register allocation phase.

After performing phase-1and phase-2 of steganography, the Pc metric is evaluated as follows:

$$Pc^2 = \left(1 - \frac{1}{c}\right)^{f1} * \left(1 - \frac{1}{\pi_{i=1}^{n} N(Ri)}\right)^{f2} \tag{6}$$

where, Pc^2 represents the probability of coincidence post phase-1 and phase-2 steganography, f2 denotes the number of effective stego-constraints (effective number of 1s) embedded in the hardware (FU) allocation phase of HLS. Further, n indicates the total types of FU (e.g. two in proposed approach), N indicates the number of functional units of type Ri. In the proposed work, N(R1) indicates the number of adders and N(R2) indicates the number of multipliers.

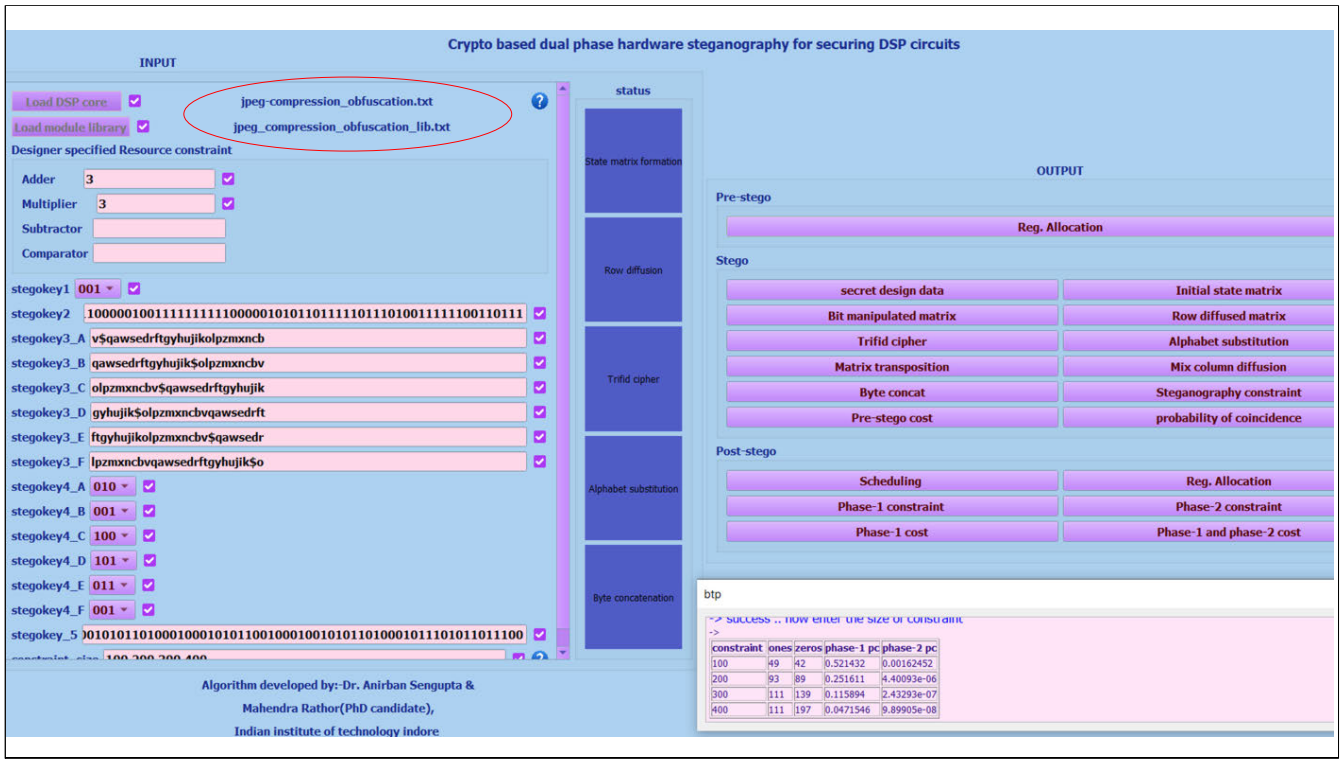
For proposed dual phase (phase-1 and phase-2) steganography, the Pc metric is evaluated for different size of stego-constraints. The effective constraints embedded increase with the increasing size of stego-constraints for both phases of steganography. For each design solution, Table 9 shows the increments in both effective number of 0s (stego-constraints for phase-1) and effective number of 1s (stego-constraints for phase-2) with increasing size of total stego-constraints (designer selected bit-stream) from 100 to 400. Further, Table 10 presents the comparison of Pc^1 and Pc^2 for different design solutions of JPEG compression processor. For each design solution, the Pc^1 and Pc^2 metrics are compared for varying size of stego-constraints (total number of 0s and 1s). As shown in the table, the value of Pc^1 and Pc^2 reduces with the increasing size of stego-constraints. This is because, as the size of stego-constraints increases, the effective number of 0s (f1) and the effective number of 1s (f2) increases, therefore the value of Pc^1 and Pc^2 reduces according to (5) and (6). Thus, by choosing the large size of stego-constraints, lower value of probability of coincidence can be achieved which in turn signifies the higher strength of steganography. Additionally, the chosen design solution (resource constraints) also impacts the Pc metric as shown in Table 10. This is because in a specific size of stego-constraints, the number of 0s and 1s vary according to chosen design solution (this can be observed in the Table 9). Therefore, the selection of appropriate design solution is very crucial to obtain higher strength of steganography. As shown in Table 10, the design solution (3+, 3*) gives the lower Pc (higher strength of steganography) than other solutions.

B. LOW COST ROBUST SECURITY OF JPEG CODEC PROCESSOR: RESULTS

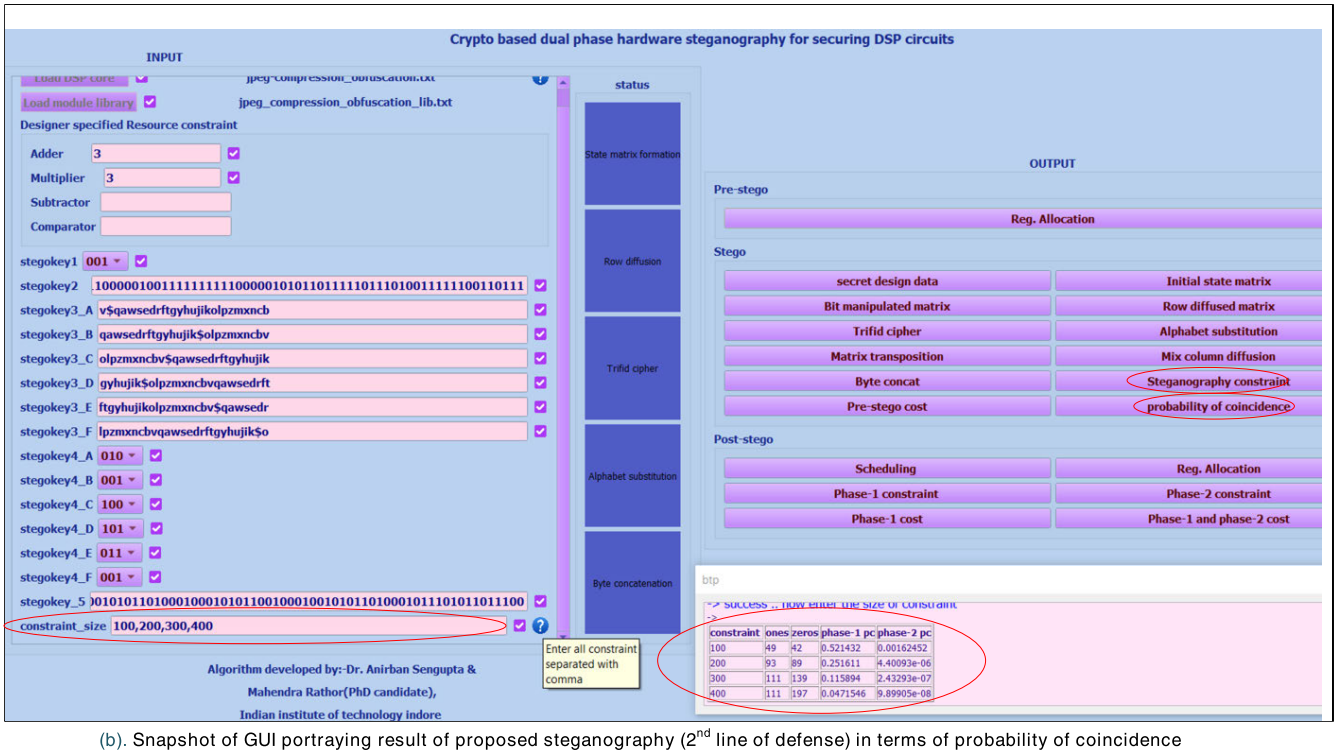
The proposed approach achieves security of JPEG compression processor at very low design cost. The design cost of pre and post steganography embedded obfuscated JPEG compression processor is evaluated based on following cost function [10]:

$$C_d\,(X_i) = w_1 \frac{L_T}{L_m} + w_2 \frac{A_T}{A_m} \tag{7}$$

where, $C_d\,(X_i)$ is the cost with resource configuration X_i, A_T and L_T denote the JPEG compression design area and delay, Am and Lm denote the maximum area and delay of the design, w_1 and w_2 are the user defined weights for latency and area respectively. Both weights w_1 and



(a). Snapshot of GUI of proposed steganography (2nd line of defense) highlighting JPEG compression processor as input



(b). Snapshot of GUI portraying result of proposed steganography (2nd line of defense) in terms of probability of coincidence

FIGURE 22. (a). Snapshot of GUI of proposed steganography (2nd line of defense) highlighting JPEG compression processor as inpu. (b). Snapshot of GUI portraying result of proposed steganography (2nd line of defense) in terms of probability of coincidence.

w_2 are fixed at 0.5 to assign equal preference. To evaluate the area and the latency of the JPEG compression design, technology scale based on NanGate library [18] is adopted.

Post phase-1 and phase-2 of steganography, the design cost is evaluated for different design solutions. Additionally, for each design solution, the cost is evaluated for varying size of stego-constraints. Table 12 presents the design cost post

TABLE 10. Security analysis (in terms of probability of coincidence) of proposed approach on varying size of stego-constraints for different design solutions.

Design solution	# of constraint =100		# of constraint =200		# of constraint =300		# of constraint =400	
	Pc^1	Pc^2	Pc^1	Pc^2	Pc^1	Pc^2	Pc^1	Pc^2
3+, 3*	5.21 e-1	1.6245e-3	2.5161e-1	4.4e-6	1.1589e-1	2.4e-7	4.715e-2	9.89e-8
3+, 5*	4.75e-1	1.732e-2	2.222e-1	3.39e-4	1.008e-1	2.228e-5	4.296e-1	9.497e-6
5+, 5*	4.68e-1	6.329e-2	2.188e-1	4.913e-3	9.328e-2	5.907e-4	3.976e-2	2.518e-4
7+, 9*	5.38e-1	2.092e-1	2.595e-1	4.61e-2	1.233e-1	1.515e-2	5.59e-2	6.87e-3
9+, 9*	5.06e-1	2.552e-1	2.364e-1	6.496e-2	1.106e-1	2.146e-2	5.42e-2	1.051e-2
11+, 11*	4.40e-1	3.001e-1	1.845e-1	8.816e-2	8.368e-2	2.821e-2	3.458e-2	1.166e-2

TABLE 11. Design cost analysis of proposed approach on varying size of stego-constraints for different design solutions.

Design solution	Pre-steganography cost	# of constraint= 100		# of constraint= 200		# of constraint= 300		# of constraint= 400	
		Cost (Phase-1)	Cost (Phase-1 & Phase-2)	Cost (Phase-1)	Cost (Phase-1 & Phase-2)	Cost (Phase-1)	Cost (Phase-1 & Phase-2)	Cost (Phase-1)	Cost (Phase-1 & Phase-2)
3+, 3*	0.2167	0.2167	0.2167	0.2167	0.2169	0.2167	0.2173	0.2167	0.2173
3+, 5*	0.1917	0.1917	0.1920	0.1917	0.1924	0.1917	0.1929	0.1917	0.1929
5+, 5*	0.1713	0.1713	0.1713	0.1713	0.1713	0.1713	0.1719	0.1713	0.1719
7+, 9*	0.1718	0.1718	0.1720	0.1718	0.1725	0.1718	0.1729	0.1718	0.1729
9+, 9*	0.1752	0.1752	0.1754	0.1752	0.1757	0.1752	0.1763	0.1752	0.1763
11+, 11*	0.1785	0.1785	0.1785	0.1785	0.1789	0.1785	0.1794	0.1785	0.1794

TABLE 12. Comparison of design cost and security of JPEG DCT core using proposed approach and [21].

Comparison metric	Proposed	[21]
Design cost	0.45	0.45
Security in terms of key size (in bits)	610	0

embedding phase-1 and phase-2 steganography and compares with the pre-steganography design cost. As shown in the table, the design cost post phase-1 steganography remains same as pre-steganography cost. This is because, embedding stego-constraints (0s) during phase-1 steganography does not require any extra register. However, post embedding phase-2 steganography, the design cost either slightly increases or remains unchanged with increasing size of stego-constraints. This is because, the embedded stego-constraints (1s) during phase-2 steganography may require more FU (adders and multipliers) allocation of vendor type-2 than that of type-1. Since, the area and latency of FUs of chosen vendor type-2 is slightly higher with respect to vendor type-1, therefore the design cost may marginally increase post phase-2 steganography. Furthermore, the chosen design solution (resource constraints) also impacts the design cost as shown in Table 11. The design cost reduces as resource constraints are increased from (3+, 3*) to (5+, 5*). This is because, increasing resource constraints upto (5+, 5*) reduces the design latency significantly with marginal increase in the design area. This results into reduction in overall design cost. However, further increase in resource constraints from (5+, 5*) to (11+, 11*) does not cause significant reduction in the design latency. This is because, the utilized number of resources (adders/multipliers) in a control step (during scheduling) are lesser than the given resource constraints. Therefore, significant reduction in design latency with respect to increase in the design area does not take place. Hence, design cost starts increasing from (5+, 5*) onwards as shown in Table 11.

Furthermore, Table 12 shows the comparison of security and design cost of JPEG DCT core using proposed approach and related work [21]. As shown in the table, the proposed approach offers higher security in terms of key size (610 bits) at zero cost overhead. Larger key size in proposed approach enhances the difficulty level for an attacker in determining the embedded stego-constraints.

Graphical user interface (GUI) of the implementation of the proposed approach is shown in Fig. 22 which portrays the inputs and results obtained for 2^{nd} line of defense (proposed steganography). Obfuscated JPEG compression processor, the input to proposed steganography, is encircled in Fig. 22(a). Further, the constraint size, steganography constraints (number of 0s and 1s) and corresponding Pc values (same as reported in Table 10) for design solution "3*, 3+" are encircled in Fig. 22(b). Similarly, results in terms of design cost can be shown.

VI. CONCLUSION

The JPEG CODEC processor used for compression and reconstruction in medical imaging systems is vulnerable to hardware threats such as malicious logic insertion, counterfeiting/cloning etc. To ensure the correctness of the medical image data, security of the JPEG CODEC processor is very crucial. In this paper, double line of defense is proposed to secure the JPEG processor. To do so, THT based structural obfuscation is used as 1^{st} line of defense and hardware steganography is embedded into the design as 2^{nd} line of defense. The embedding of proposed dual phase steganography on the top of the obfuscated JPEG design enables

robust security of the JPEG processor design against aforementioned hardware threats for medical imaging systems. The proposed approach provides higher security at negligible design cost as shown in the results.

REFERENCES

[1] D. A. Koff and H. Shulman, "An overview of digital compression of medical images: Can we use lossy image compression in radiology?" *Can. Assoc. Radiol. J.*, vol. 57, no. 4, pp. 211–217, 2006.

[2] S. Gokturk, C. Tomasi, B. Girod, and C. Beaulieu, "Medical image compression based on region of interest, with application to colon CT images," in *Proc. Conf. 23rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, vol. 3, Aug. 2005, pp. 2453–2456.

[3] S. B. Gokturk. *Region of Interest Based Medical Image Compression*. Accessed: Aug. 2019. [Online]. Available: http://ai.stanford.edu/~gokturkb/Compression/FinalReport.htm

[4] Y.-Y. Chen and S.-C. Ti, "Embedded medical image compression using DCT based subband decomposition and modified SPIHT data organization," in *Proc. 4th IEEE Symp. Bioinf. Bioeng.*, Oct. 2004, pp. 167–174.

[5] Y.-Y. Chen, "Medical image compression using DCT-based subband decomposition and modified SPIHT data organization," *Int. J. Med. Inform.*, vol. 76, no. 10, pp. 717–725, Oct. 2007.

[6] R. Agarwal, C. S. Salimath, and K. Alam, "Multiple image compression in medical imaging techniques using wavelets for speedy transmission and optimal storage," *Biomed. Pharmacol. J.*, vol. 12, no. 1, pp. 183–198, Mar. 2019.

[7] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware trojans in third-party digital IP cores," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust*, San Diego CA, USA, Jun. 2011, pp. 67–70.

[8] Maxim. Accessed: May. [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/545

[9] SMT Corp. *Counterfeit Detection*. Accessed: May 2019. [Online]. Available: https://www.smtcorp.com/counterfeit-detection

[10] A. Sengupta, D. Roy, and S. P. Mohanty, "Triple–phase watermarking for reusable ip core protection during architecture synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 4, pp. 742–755, Apr. 2018.

[11] A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, "DSP design protection in CE through algorithmic transformation based structural obfuscation," *IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 467–476, Nov. 2017.

[12] Y. Lao and K. K. Parhi, "Obfuscating DSP circuits via high–level transformations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 5, pp. 819–830, May 2015.

[13] A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, "Low–cost obfuscated JPEG CODEC IP core for secure CE hardware," *IEEE Trans. Consum. Electron.*, vol. 64, no. 3, pp. 365–374, Aug. 2018.

[14] A. Sengupta, R. Sedaghat, and Z. Zeng, "A high level synthesis design flow with a novel approach for efficient design space exploration in case of multi-parametric optimization objective," *Microelectron. Rel.*, vol. 50, no. 3, pp. 424–437, Mar. 2010.

[15] M. C. McFarland, A. C. Parker, and R. Camposano, "Tutorial on high-level synthesis," in *Proc. 25th ACM/IEEE Design Autom. Conf. (DAC)*, Hoboken, NJ, USA, Jun. 1988, pp. 330–336.

[16] A. Sengupta and S. Bhadauria, "Exploring low cost optimal watermark for reusable IP cores during high level synthesis," *IEEE Access*, vol. 4, pp. 2198–2215, 2016.

[17] F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 3, pp. 523–545, Jul. 2005.

[18] *NanGate 15 nm Open Cell Library*. Accessed: Jun. 2019. [Online]. Available:http://www.nangate.com/?pageid=2328

[19] Kegal. *CT Medical Images*. Accessed: Jul. 2019. [Online]. Available: https://www.kaggle.com/kmader/siim-medical-images/home

[20] A. Sengupta and M. Rathor, "Protecting DSP kernels using robust hologram–based obfuscation," *IEEE Trans. Consum. Electron.*, vol. 65, no. 1, pp. 99–108, Feb. 2019.

[21] A. Sengupta and M. Rathor, "IP core steganography for protecting DSP kernels used in CE systems," *IEEE Trans. Consum. Electron.*, vol. 65, no. 4, pp. 506–515, Nov. 2019.

[22] B. Le Gal and L. Bossuet, "Automatic low-cost IP watermarking technique based on output mark insertions," *Des. Autom. Embedded Syst.*, vol. 16, no. 2, pp. 71–92, Jun. 2012.

[23] A. Sengupta and D. Roy, "Anti-piracy aware IP chipset design for CE devices: Robust watermarking approach," *IEEE Consum. Electron. Mag.*, vol. 6, no. 2, pp. 24–118, 2017.

ANIRBAN SENGUPTA (Senior Member, IEEE) is currently an Associate Professor in computer science and engineering with IIT Indore, where he directs the research lab on "CAD for Consumer Electronics Hardware Device Security and Reliability." He has more than 214 publications, including three Books and 11 Patents. He is the author of three Books from the IET and Springer on Hardware Security, IP core protection, and VLSI Design. His patents have been used and cited in industry patents/products of IBM Corporation, Siemens Corporation, Qualcomm, Amazon Technologies, Siemens (Germany), Mathworks Inc., Ryerson University, and STC University of Mexico multiple times. He is an elected Fellow of the IET and a Fellow of the British Computer Society (FBCS), U.K. He has been awarded the prestigious IEEE Distinguished Lecturer by the IEEE Consumer Electronics Society, in 2017, and the IEEE Distinguished Visitor by the IEEE Computer Society, in 2019. He is a recipient of several IEEE Honors such as the IEEE Chester Sall Memorial Consumer Electronics Award 2020, the IEEE Outstanding Editor Awards, the IEEE Outstanding Service Awards, and the IEEE Best Paper Awards from Journals/Magazines and Conferences. He was the General/Conference Chair of the 37th IEEE International Symposium on Consumer Electronics (ICCE) 2019, the Las Vegas and Technical Program Chair of the 36th IEEE International Conference on Consumer Electronics (ICCE) 2018 in Las Vegas, the 9th IEEE International Conference on Consumer Electronics (ICCE) - Berlin 2019, the 15th IEEE International Conference on Information Technology (ICIT) 2016, and the 3rd IEEE International Symposium on Nanoelectronic and Information Systems (iNIS) 2017. He is currently the Deputy Editor-in-Chief of the *IET Computers and Digital Techniques Journal* and the Editor-in-Chief of the IEEE VLSI Circuits & Systems Letter of the IEEE Computer Society TCVLSI. He is currently the Chairman of the IEEE Computer Society TCVLSI. He currently serves/served in more than 16 Editorial positions of several IEEE Transactions/Journals, IET, and Elsevier Journals, including the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS (TAES), the IEEE TRANSACTIONS ON VLSI SYSTEMS, the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, IEEE ACCESS Journal, the *IET Journal on Computer & Digital Techniques*, the IEEE CONSUMER ELECTRONICS, the IEEE CANADIAN JOURNAL OF ELECTRICAL AND COMPUTER ENGINEERING, the IEEE VLSI CIRCUITS AND SYSTEMS LETTER, and other Journals. He is a registered Professional Engineer of Ontario (P.Eng.).

MAHENDRA RATHOR (Member, IEEE) received the M.E. degree in electronics engineering, in 2014. He is currently pursuing the Ph.D. degree in computer science and engineering with IIT Indore.