A Secure Digital Camera Architecture for Integrated Real-Time Digital Rights Management

Saraju P. Mohanty
VLSI Design and CAD Laboratory (VDCL)
Dept. of Computer Science and Engineering
University of North Texas, Denton, TX 76203, USA.

Abstract

This paper presents a novel concept of a secure digital camera (SDC) with a built-in watermarking and encryption facility. The motivation is to facilitate real-time digital rights management (DRM) by using the SDC right at the source end of multimedia content. The emerging field of DRM systems addresses the issues related to the intellectual property rights of digital content. The use of digital watermarking along with encryption for effective DRM is proposed. In this context, a novel discrete cosine transform domain invisible-robust watermarking method that uses cryptography and watermarking methods simultaneously to provide a double layer of protection to digital media is presented. The proposed method securely hides binary images in color image media and securely extracts and authenticates it by using a secret key. Experimental results prove that the proposed technique is resilient to stringent watermarking attacks. Hence, it is an effective method for providing protection of ownership rights. The corresponding application-specific architectures for invisible-robust watermarking and Rijndael advanced encryption standard (AES) towards the prototyping of the SDC are presented. The proposed architectures are modeled and synthesized for field programmable gate array (FPGA). The soft cores in the form of hardware description language resulting from this research can serve as intellectual-property core and can be integrated with any multimedia-producing electronic appliances which are built as embedded systems using system-on-a-chip (SoC) technology.

Key words: Application Specific Architectures, VLSI (Very Large Scale Integration) Architectures, Watermarking Chip, Digital Rights Management (DRM), Copyright Protection, Multimedia Security, Digital Watermarking, Encryption, Real-Time DRM

1. Introduction and Motivation

The Internet revolution toward the end of the last millennium ushered in a new era of information technology. There has been an explosive growth in multimedia applications such as video-on-demand, distance education, etc. Flexibility to avail digital content in form of images and video has many issues. Easy access facilitates information piracy through unauthorized replication and manipulation of digital content with the help of inexpensive tools. Hence, concerns about protection and enforcement of intellectual property (IP) rights of the digital content involved in transactions have also been mounting. The emerging field of digital rights management (DRM) systems [1,2] addresses these issues related to ownership rights of digital content.

Email address: saraju.mohanty@unt.edu (Saraju P. Mohanty).

Various aspects of content management - namely, content identification, storage, representation, distribution and intellectual property rights management - are highlighted in DRM [1,2]. Unauthorized access of digital content is being prevented by implementing encryption technologies, but their implementation does not prevent illegal replication of the decrypted content by an unauthorized user. Hence, encryption alone does not address all the IP issues related to DRM. Digital watermarking can be used for establishing ownership rights, tracking usage, ensuring authorized access, preventing illegal replication, and facilitating content authentication. In essence, digital watermarking is the process of embedding into a multimedia object a digital signature or data that is variously known as watermark, tag, or label. Detection or extraction of this watermark at a later time enables users to make an assertion about the authenticity and ownership of the object. When encryption techniques are used with watermarking, full protection from unauthorized access of digital content can be achieved. The novel approach proposed in this paper aims at providing two-tier protection mechanism with simultaneous use of cryptography and watermarking.

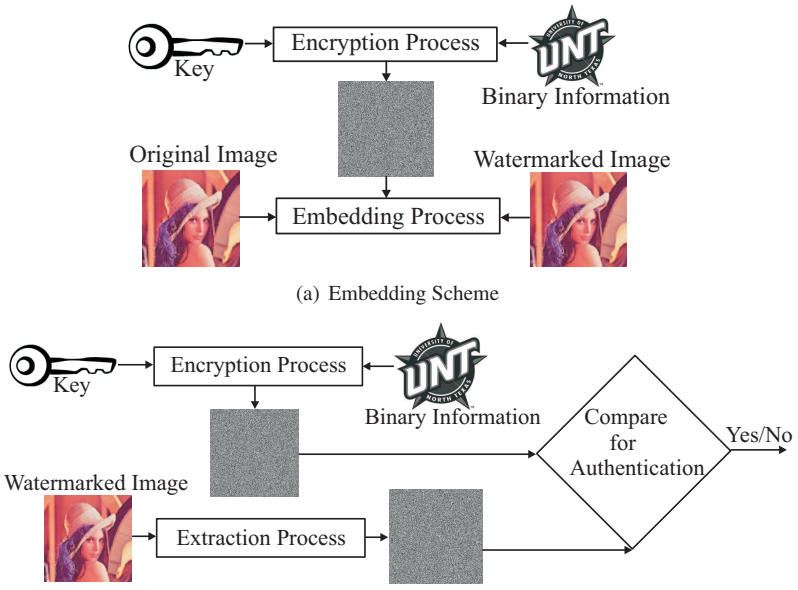
Research on watermarking has matured over the last decade; hence, the current literature abounds with techniques in this area [3,4,5,6,7,8,9]. Based on human perception, digital watermarks can be divided into visible and invisible (robust or fragile) types [10,11,12]. These existing algorithms primarily work offline because they are computationally very intensive and cannot be applied in real time. However, the security and copyright protection mechanisms have to work in real time in the case of emerging applications; namely, digital television broadcasting, Internet protocol television (IP-TV), video on demand, pay-TV, electronic passport (e-passport), credit cards, personal identity cards, driving licenses, etc. Consequently, appliances such as digital still cameras, digital motion cameras, network processors, mobile phones, video phones, graphics processing units, and digital video disc (DVD) players need to be equipped with mechanisms for real-time security and copyright protection. In these situations, software-only solutions may not be adequate to provide real-time performance. *Hardware-assisted solutions are needed* for providing real-time performance along with easy integration in multimedia hardware, low-power consumption, higher reliability and availability compared to software, and low cost [13]. This need to provide real-time performance is the motivation for exploring hardware-assisted solutions for DRM in the framework of the camera, one of the most common multimedia creating consumer appliances. The system level solution presented as secure digital camera (SDC) is a real-time DRM solution which performs real-time DRM of images (or video frames) without using computers and associated specialized softwares.

The rest of the paper is organized as follows. The contributions of this paper are highlighted in Section 2. An overview of the SDC is presented in Section 3. Related research work is presented in Section 4. A discussion of the proposed algorithm is provided in Section 5. The proposed architectures are described in Section 6. The experimental results are presented in Section 7. Conclusions, future work, and sample applications are provided in Section 8.

2. Novel Contributions of this Paper

The novel contributions of this paper are follows. First, the paper introduces the concept of the secure digital camera (SDC) with built-in DRM facility which is suitable for real-time applications. SDC is a low-cost and high-performance watermarking solution which performs watermarking without using separate computer and softwares. An overview is presented for the system-level architecture of SDC with the associated design challenges to promote further research in this area. Possibilities of alternative designs for the SDC are also presented. In this paper, we then focus on two key components of SDC, the watermarking and the encryption. A new algorithm, corresponding architectures, and field programmable gate array (FPGA) based prototyping for DRM are presented for one of the selected alternative designs of the SDC. The paper presents a novel invisible watermarking method called "CryptMark" that uses cryptography and watermarking methods simultaneously to provide a double-layer protection to the digital media which can be an effective technique for DRM. The method proposed securely hides binary images in color image, extracts it, and authenticates it by using a secret key as demonstrated in Fig. 1.

The advantage of the encrypted watermark processing as proposed in this paper is that at no point is raw watermark information passed in the transmission channel, thus providing maximum security. The proposed embedding process uses both direct current (DC) and alternating current (AC) discrete cosine transform (DCT) components to carry the payload, unlike most of the existing algorithms that heavily rely on low-frequency AC components. This use of DC and AC components provides more resilience to lossy compression, a process that is heavily dependent on smaller low-frequency values of AC components. In addition, selective addition or subtraction of the watermark from the



(a) Embedding Scheme

(b) Authentication Scheme

Fig. 1. The Proposed Secure Watermarking Scheme: CryptMark

DCT coefficients is performed instead of the addition operation in typical available algorithms. Thus, the proposed approach allows carrying maximum payload with the highest robustness and highest undetectability, which are three contradictory objectives of data hiding mechanisms. This is evident from Section 5 by the fact that the algorithm has been able to hide watermark images completely inside the host images and produce high quality watermarked images (high peak-signal-to-noise ratio, $PSNR$) compared existing similar algorithms.

In this paper, novel architectures for the invisible watermarking [14] and Rijndael Advanced Encryption Standard (AES) [15] that will be integrated in the SDC are presented. The approaches of parallelism and resource-sharing are used to meet the timing and area constraints. The architecture for AES is area-optimal because the round key for each transformation is calculated on the fly, instead of storage of all calculated round keys. The FPGA prototype version of the proposed watermarking architecture is estimated to have a maximum operating frequency of $256MHz$ and the encryption architecture is estimated to have a maximum operating frequency of $204MHz$.

3. The Proposed Secure Digital Camera (SDC)

3.1. *SDC: A System-Level Perspective*

The SDC is an appliance built as a system on a chip (SoC) with the standard features of a digital camera and a built-in facility for real-time, low-cost, and low-power DRM. For a given multimedia data (image or video), SDC needs to prove many DRM-related tasks, including: (i) copyright information (using visible and invisible-robust watermarking); (ii) extent of tampering (invisible-fragile watermarking); (iii) source of image - camera information, place, or date (invisible-robust or visible watermarking); (iv) owner's, creator's, or cameraman's information (invisible-robust or visible watermarking), etc. An algorithm, an architecture, and a chip for visible watermarking have been presented in our previous publication [16]. This paper focuses on the invisible-robust watermarking algorithm and architectures.

The system-level block diagram of the proposed SDC is shown in Fig. 2. It identifies most of its components, such as active pixel sensor (APS) unit, liquid crystal display (LCD), memory, encryption unit, compression unit, bar code unit, and watermarking unit. In the proposed SDC, the image is captured by an image sensor and converted to a digital signal by the analog-to-digital converter (ADC). A complementary metal oxide semiconductor (CMOS) image sensor that has an embedded ADC is used. The captured image is stored temporarily in the scratch memory, after which it is displayed on the LCD panel with the help of the controller. The scratch memory can either be a volatile or non-volatile

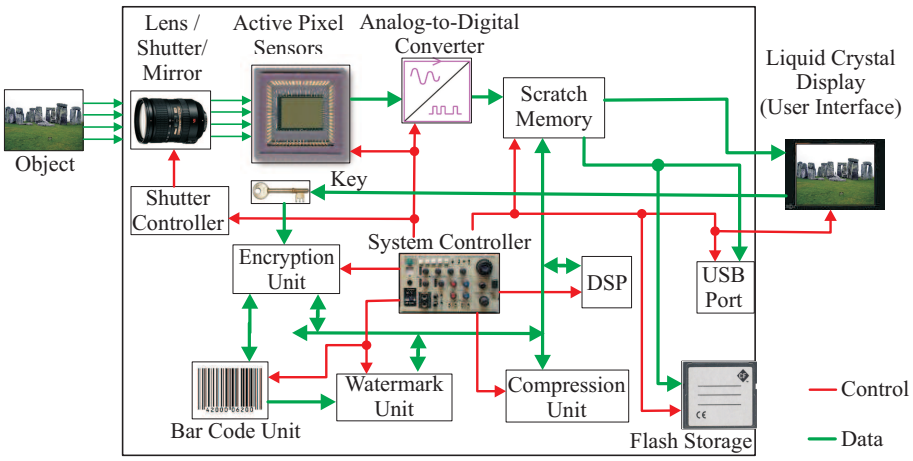


Fig. 2. The Proposed SDC: A System-Level Representation

memory. The purpose of the LCD panel is to enable the user to see the image frame before it is watermarked by the watermarking unit and stored in the camera. The image can then be further transmitted over the network or transferred to flash memory, computer hard drive, or optical discs. The flash memory is a non-volatile memory used for permanent storage of images in camera. The controller unit is responsible for controlling the entire sequence of events. Both the invisible-robust and visible watermarking algorithms can be used along with encryption and data compression for different purposes. The choice of operations performed on the image depends on the user of the camera. This paper focuses on the structural aspects of the controller, watermarking, and encryption units to develop the prototype of the complete SDC.

It may be noted that the system presented for SDC is a generic context. The exact architecture depends on specific applications. The application will determine the followings: (i) the selection of exact watermarking, encryption, and compression algorithm, (ii) the architectures of corresponding individual modules of SDC, such as watermarking, encryption, compression, and (iii) the amount of on-chip memory. We now discuss some representative examples. For an application like photo-ID card, the watermarking algorithm may be an visible watermarking and the watermarked photos needed to be stored in flash storage. For a live broadcasting applications (such as live sports), both visible and invisible-robust blind watermarking is needed. The visible watermark is a broadcasters' logo and invisible-robust blind watermarking is a permanent copyright protection mechanism at the top of it. For high-quality image watermarking applications the owner wants to have original image with him, but wants to release only watermarked images. In this case, invisible-robust non-blind watermark is most suitable for application point of view. SDC system point of view, the amount of flash memory needs to be maximum to store both original and watermarked images. However, with the cost of memory coming down and high-capacity solid-state memory available in the market, this is not an issue from either cost, technology, or system power consumption. Invisible-robust non-blind watermarking is needed for biometric based watermarking which finds application in e-passport applications [17]. In this paper, we focus on AES encryption and invisible-robust non-blind watermarking. The future research in this direction is for invisible-robust blind watermarking and their low-power, low-cost architectures.

3.2. *Motivation for Hardware-Assisted DRM Through SDC*

The *advantages of the proposed hardware-assisted DRM* over existing software-based DRM are as follows:

(i) Hardware-based DRM can be easily integrated with multimedia hardware, such as digital cameras, network processors (NPs), graphics processing units (GPUs), etc. However, this paper focuses only on digital cameras.

(ii) Power consumption for hardware-based DRM is lower compared to that of software because DRM would not need to use the power-hungry general-purpose processor. Hence hardware-based DRM is more suitable for battery operated embedded systems.

(iii) Performance in hardware-based DRM is higher compared to that of software since hardware can be custom-built for high throughput.

(iv) Hardware has higher reliability and availability compared to software; as softwares may have bugs, softwares need maintenance, software is also affected by the performance of the hard disk or system in which they are parked for permanent storage.
(v) Hardware-based DRM is absolutely needed for real-time applications like digital video broadcasting. Software-based DRM simply cannot be used in this case to provide real-time performance processing high-resolution video frames at the rate of typically 29.97 frames per second.
(vi) The cost is low compared to having explicit software because it can be monolithically built on a single unified system in the context of SoC technology in cameras, NPs, or GPUs, with minimal hardware overhead when compared to a system without DRM.
(vii) Hardware-based DRM right at the source e.g. at sensors or analog-to-digital converter (refer next subsection) ensures that the information is always protected.
(viii) Hardware-based DRM integrated with multimedia creating or processing components (digital signal processor, cameras, NPs, or GPUs) rather than separate software running in a separate computer will be more acceptable as legal evidence.

It may be noted one misapprehends that if security features are implemented in hardware platforms then they can not be modified later. There are several aspects for this issue. First, the primary motivation for hardware assisted solution is real-time performance and performing the operations without using softwares in a PC for this task, which leads to offline approaches. The second thought is that to maintain upgradability of the hardware one can implement the hardware as a soft core expressed in structural Verilog or VHDL hardware description language. The soft core can be modified as algorithm changes and resynthesized to build new silicon.

3.3. *The Proposed Design Alternatives*

In the system-level design perspective of SDC, the following alternative approaches for designing the SDC are proposed. It is envisioned that they will give rise to multi-front research in DRM including from the analog VLSI design and the embedded system design researchers.

3.3.1. *New CMOS Sensor with DRM*

Recently, a completely new generation of image sensors, the digital pixel sensor (DPS), has been getting lots of attention. In a DPS, each pixel consists of a photo-detector, an ADC, and a digital memory for temporary storage of data before a digital output signal is read out. This pixel array can be accessed randomly from digital memory provided. DPS has several advantages over analog image sensors, such as passive pixel sensor (PPS) and APS [18]. DPS provides better scaling with CMOS technology and reduces fixed pattern noises and readout noises. Thus, integration of pixel-level watermarking processing is an attractive option with minimal overhead [19]. The watermark inserted at this stage of the multimedia will provide maximum security.

3.3.2. *New ADC with DRM*

Another alternative for sensor-end watermarking of the multimedia information is to design a new type of ADC. This option can be useful in the case of APS, PPS, and charge coupled devices (CCDs). In these cases, ADC is a distinct component contrary to DPS [20]. Thus, integrating pixel-level watermarking processing in the circuit of ADC is an lucrative option with minimal overhead to provide security to multimedia content for a digital camera irrespective of the type of image sensor.

3.3.3. *Independent DRM Processors*

In this design option, physically individual processors for invisible watermarking, visible watermarking, and encryption can be designed as intellectual property (IP) cores and integrated in the framework of an SoC realizing the SDC. This design is currently the most attractive one and can potentially provide the highest performance. Moreover, this approach can work with any type of digital camera as demonstrated in this paper. In this method, cores can be switched off when not in use by clock gating, resulting in better power dissipation [16]. We follow this design alternative in this paper.

3.3.4. *DRM coprocessor for DSP*

In a typical digital camera, the main work horse is a digital signal processor (DSP) that performs primitive image processing operations. So the fourth design alternative is to modify the design of the DSP to have integrated co-processors to perform DRM. However, this modification can potentially affect the operating frequency of the DSP and subsequently overall performance.

3.3.5. *New instruction set architecture for a reduced instruction set computer (RISC) to support DRM at micro-architecture level*

When designed as an embedded system with a DSP or application-specific integrated circuit (ASIC) of the RISC architecture being the main high-performance component, this option can be attractive [21]. The instruction set architecture of the RISC component can be modified to have explicit encryption and watermarking as instructions. This modification can be beneficial when these instructions are used quite frequently. The potential downside of this approach is performance overhead.

4. Related Prior Research

Research on digital rights management is in full swing and several results have been presented in current literature. In this section we discuss the DRM research which are hardware assisted solutions and present them in the context of digital camera. The trustworthy camera, with the aim of restoring credibility to photographic images using encryption, is presented in [22]. This is the earliest attempt that have incorporated cryptography in the camera. The development of a biometric authentication system for a secure camera is discussed in [23]; however, application specific very-large-scale integration (VLSI) architecture was not proposed. However, the term of "secure digital still camera" was coined by us for first time in the literature in [24] and a visible watermarking chip was presented there.

Additionally, several watermarking chips and FPGA prototypes are presented in the literature. The authors in [25] propose a real-time watermarking algorithm. This algorithm is extended to a VLSI chip in [26]. A DCT domain invisible watermarking chip is presented in [27]. The authors in [28] propose a watermarking VLSI architecture for invisible fragile watermarking. A visible watermarking design based on an adaptive discrete wavelet transform is presented in [29]. The authors in [30] present a design for a CMOS APS imager incorporating circuits for a pseudo-random generator for invisible watermarking. In [31], both semi-fragile and robust watermarks are employed when images are captured by digital camera, but application specific VLSI or hardware architecture was not presented. In [32], concept of single-sensor digital camera that inserts visible watermarks is presented; this also does not deal with VLSI or hardware architecture.

In this paper, architecture for an SDC with both watermarking and encryption for image security and authentication is introduced as an advancement of the state-of-the-art. A new algorithm that uses both encryption and watermarking is presented which is based on our conference presentation [14]. The corresponding VLSI architectures are proposed and prototyped with Xilinx FPGA based on the preliminary results presented in our conference publication [33].

5. The Proposed Algorithm : CryptMark

5.1. *The Embedding Algorithm*

The algorithmic flow of secure insertion process of the proposed watermarking is represented in Fig. 3. The algorithm first encrypts the watermark and then fuses it into the intensity image of a grayscale cover image or into the Y-component (in the YC_rC_b coordinate system) of a color image. The relevant component of the cover image is referred as I. The image is decomposed in the preprocessing stage of the algorithm to obtain the required component. The binary watermark is encrypted with a user-supplied key in this preprocessing step. At this step, any image extension necessary to facilitate the division of the image into integral number of blocks is performed.

After preprocessing, the cover image I is divided into 8×8 blocks and each block is transformed into the DCT domain. The "(i, j)"th DCT coefficient of the kth block is denoted by $c_{ij}(k)$. Suppose that the image has M blocks overall; each block can be numbered uniquely with a number in the range $[1, M]$ based on its position in the raster

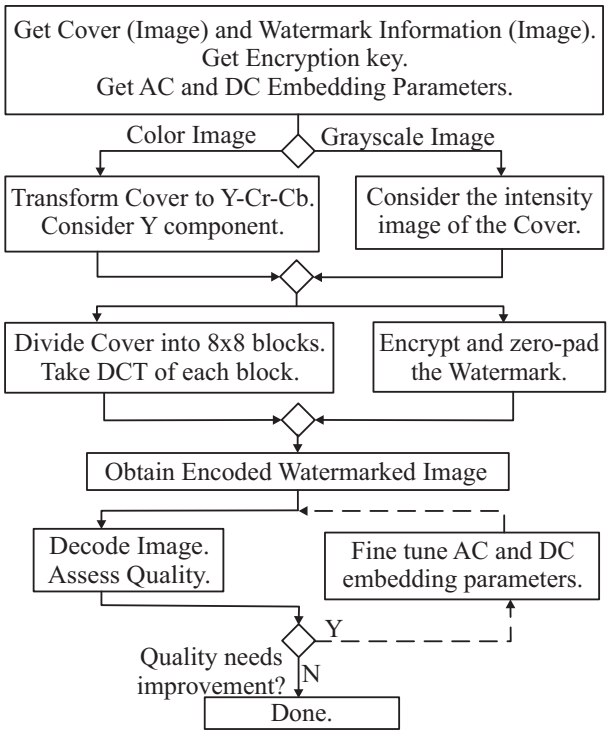


Fig. 3. The Proposed Algorithmic Flow of the Insertion Process.

scanning of the image. M is given by $(ROW \times COL/64)$, where ROW is the number of image pixels row-wise, and COL is the number of pixels column-wise. Based on experience with perceptual analysis, the number of DCT frequency components has to be decided to obtain high-quality watermarked images.

Let us assume that only the DC component c_{00} is needed and the three low frequency components c_{01}, c_{10}, and c_{11}. In this case, the size of the encoded and hashed watermark should be such that it can be partitioned into the same number of blocks as the cover image, but with a block size of 2×2. It cannot be bigger; but if it is smaller, it can be padded with zeros. Let us now denote the watermark's binary value at position (i, j) in block k by $w_{ij}(k)$. This watermark is embedded in the cover image by using the following expression: $\forall i, j, k$,

$$c^*_{ij}(k) = \begin{cases} c_{ij}(k)(1 + \alpha_{ij}) \;\; if \;\; w_{ij}(k) = 1, \\ c_{ij}(k)(1 - \alpha_{ij}) \;\; if \;\; w_{ij}(k) = 0. \end{cases} \tag{1}$$

Unlike the method in [3], in the current approach, the watermark is not always added to the significant frequency components. Instead, it is added to some components and subtracted from the other components [4]. This action strengthens the requirement that a statistical analysis of the watermarked image should not reveal the presence of an invisible watermark. Unlike the method in [3], four embedding factors are used: α_{DC} for DC components and α_{AC} for AC components. Thus, $\alpha_{00} = \alpha_{DC}$ and $\alpha_{01} = \alpha_{10} = \alpha_{11} = \alpha_{AC}$. Because choosing so many scaling factors (one for each frequency component) is a optimization problem in itself, only two values have been chosen so as not to degrade the quality of the watermarked image. Image quality can be assessed either quantitatively by measuring its peak-signal-to-noise ratio $(PSNR)$ or other similar measures.

In the case of grayscale cover images, the watermarked image I^* can be obtained by performing block-wise inverse DCTs (IDCTs) on the coefficients modified as above. However, in the case of colored images, only the Y-component of I^* is acquired by the above process. The Y-component is clubbed with the C_r and C_b components of the cover to get I^*. At this point, an optional step (dashed line in Fig. 3) of assessing quality of the watermarked image may be executed by a quality measure and fine-tuning of the parameters α_{AC} and α_{DC}.

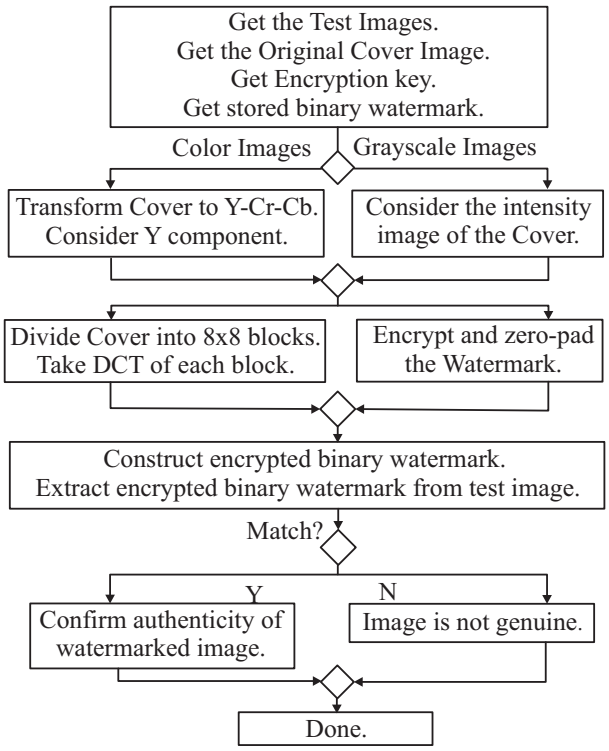


Fig. 4. The Proposed Algorithmic Flow of the Extraction and Authentication Process.

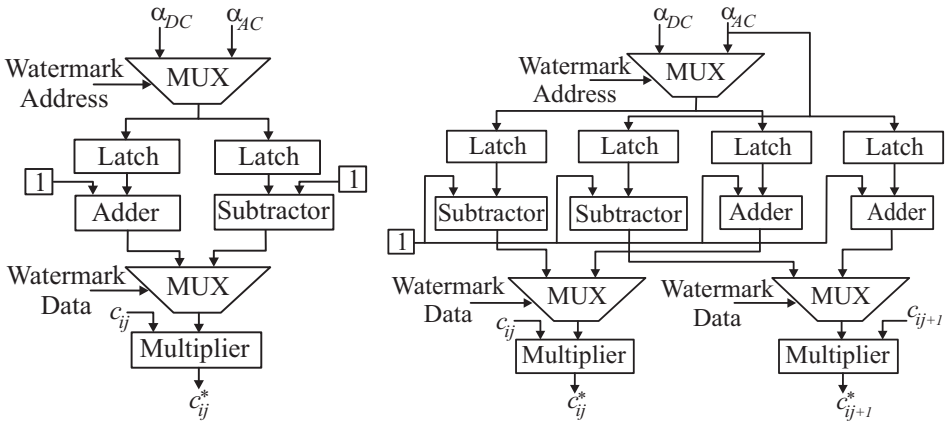
5.2. *The Extraction and Authentication Algorithm*

The flow of secure extraction and authentication process in the proposed CryptMark is demonstrated in Fig. 4. The extraction algorithm involves the following sequence of steps. First, the watermarked (possibly suspect) test image I^+ and the original cover image I are obtained. The watermark information (image) and the original encryption key are then obtained. After initial preprocessing, both I^+ and I are divided into 8×8 blocks. During this phase, if the image is color, then it is converted from RGB space to YC_rC_b representation. DCT coefficients of both the images are obtained for all the blocks. The blocks of both test image and original image are then compared. If a DCT coefficient in a block of I^+ is larger than the corresponding coefficient in the original image block, then the watermark bit is 1; otherwise, it is 0. Finally, the extracted sequence with the binary watermark (encrypted with the key) is compared to make a decision on authenticity of the image.

6. The Proposed Application Specific Architectures for CryptMark

In this section, VLSI architectures for the invisible-watermarking unit and the cryptographic unit are discussed. The invisible-insertion architecture and encryption architecture are presented. The structures of the extraction and insertion modules are similar; therefore, for brevity, only the insertion module is discussed here. In same spirit, the decryption module is structurally similar to encryption module and hence not presented here. The rest of the units of the camera design are being researched and will be presented in subsequent publications. The datapath components of the proposed architectures, such as adder, subtractor, multiplier, multiplexers, etc. are designed following different approaches from [34,35,36,33].

We present new architectures of an invisible watermarking unit in Fig. 5 which will be integrated in the SDC. We used resource parallelism and resource-sharing to meet the timing, area, and performance tradeoffs. One invisible watermarking architecture is area efficient which uses minimal number of resources and other architecture is high performance. The AES architecture in Fig. 7 is high performance, high throughput, and area efficient. The proposed



(a) Area-Efficient Architecture (b) High-Performance Architecture

Fig. 5. The Proposed Datapaths of the Invisible-Robust Insertion Architecture

AES architecture is optimized for high throughput in terms of the data rates using partial pipelining [36,33]. The hardware complexity is decreased by realizing polynomial multiplication using XOR operation. Memory optimization is achieved by selective use of look-up tables and combinational logic. An important feature of the proposed AES architecture is an effective solution of real-time round key generation that requires significantly less storage for buffering, thus particularly suitable for real-time DRM through SDC.

Three block random access memories (RAMs) are needed in the architecture: one block RAM is used as a scratch memory to store an individual image, the second RAM stores the watermark, and the third RAM stores data that has undergone processing in the SDC. Registers have been used extensively in the DCT submodule to aid utilization of pipelining. An address generator serves as an address decoder by supplying the appropriate address needed for reading from the memory.

6.1. *The Proposed Architecture for Invisible Watermarking*

The architecture of the invisible watermarking insertion architecture consists of two distinct modules: insertion datapath module and controller module. The insertion module performs the watermarking insertion process. The architecture of this module, as shown in Fig. 5(a), uses a minimal number of resources. The architecture consists of one multiplier, two multiplexers (MUX), one adder, one subtractor, and two latches. The insertion unit takes the DC DCT component (c_{00}) and the first (c_{01}), second (c_{10}), and third (c_{11}) AC components of each 8×8 block for watermarking. The top multiplexer is used to choose between the watermarking strength factors, α_{AC} and α_{DC}. Then a multiplexer helps in selecting an additive or subtractive process of watermarking insertion. To improve the performance of this architecture, an alternative version is developed that uses more resources in parallel as shown in Fig. 5(b). This parallel architecture provides the capability to watermark a DCT block in two clock cycles instead of four, improving the performance of the system. However, there is a trade-off between the performance and the area used. Latches are used in the insertion module for temporary buffering.

One of the computationally intensive units needed in the datapath architecture is the DCT module. The DCT module consists of two 1D DCT sub-modules and is implemented by following a similar approach as that of our previous work [34]. The DCT module in the current paper is realized in field programmable gate array contrary to custom integrated circuit in the previous work [34]. Buffer circuitry is used to assist in finding the transpose. It also serves as temporary storage for the first 1D DCT coefficient. In order to reduce latency, a multiplexer is used between the buffer and the second 1D DCT submodule. As opposed to RAM cells, registers are typically used to design the transpose buffer for reducing latency and increasing performance. The DCT module is controlled by the main controller and has no separate controller.

The controller is modeled as a finite state machine (FSM) with eight states (init, S_0 to S_6) as shown in Fig. 6. Transition from the initial state (init) to S_0 occurs when the start signal is high. The pixels (I_{ij}) are read from storage to the input register for their DCT coefficients to be calculated. The first DCT operation is carried as a pipelined

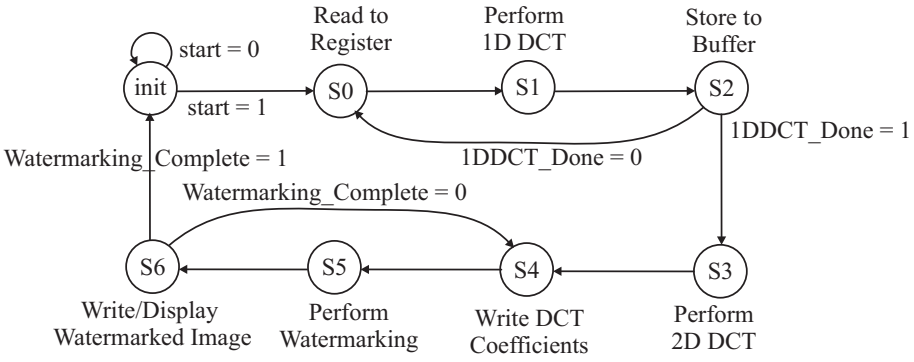


Fig. 6. Finite State Machine Presenting the Controller of Invisible Watermarking Architecture

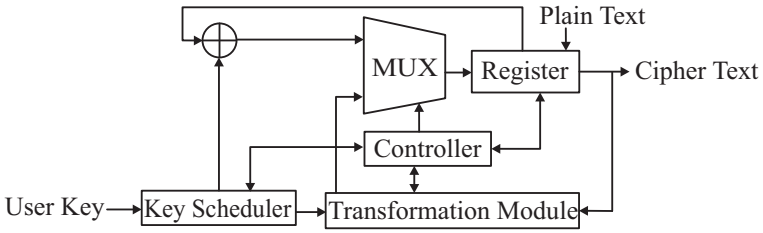


Fig. 7. High-Level Architecture of the Encryption Unit

operation. If the DCT coefficient of all the coefficients of a block is not completed, there is a transition from state S2 to state S0. After the completion of the 1D DCT operation on the original image pixel (I_{ij}) of the block, there is a transition to state S_3 for the second DCT operation. The input to the second DCT is given in parallel because of the use of the transpose buffer and the multiplexer. The 2-D DCT coefficients (c_{ij}) of the original image are obtained in state S_4. In state S_5, the process of watermarking is performed on (c_{ij}) and is then written to RAM or displayed in state S_6. If all the coefficients of the block are watermarked, a transition occurs to the initial state.

6.2. *The Proposed Architecture for the Rijndael Advanced Encryption Standard (AES)*

A high-level view of the encryption unit architecture is presented in Fig. 7. The datapath unit consists of the initial round of key addition, standard rounds, and a final round. In this paper, the implementation supports 128 bits of data and key length. The initial round operation is carried out by XORing the 128-bit plain text with the 128-bit input key. The plain text input and the key input are retrieved from the input register. The output from the initial round is then passed through a multiplexer to the register for temporary storage, after which it is then passed to the transformation module. A round key is generated for each round by the key scheduler module. The output is iterated back into the round module through the multiplexer. The transformation module is executed 10 times. The control module takes care of the sequence of operations of the encryption unit. For brevity we present a brief description of the AES architecture in this subsection and we point readers to our conference publications [36,33] for detailed understanding of the AES algorithm, FPGA prototyping, and chip implementation.

6.2.1. *The Transformation Module*

In Fig. 8, the different components of the transformation module - namely, the ByteSub, ShiftRow, MixColumn, and AddRoundkey submodules - are shown. The MixColumn submodule is used only in the standard rounds and not in the final round.

ByteSub submodule: The ByteSub submodule is made up of multiplicative inverse in $GF(2^8)$ and linear affine mapping over $GF(2)$ transformations as shown in Fig. 9. The Galois field $GF(2^8)$ is defined by finding a polynomial that is irreducible over $GF(2)$. The multiplicative inverse operation is carried out with the aid of substitution box (S-box) and affine mapping with XOR blocks. The architecture consists of 16 S-boxes operating in parallel. A “Byte

Fig. 8. Architecture of the Transformation Module

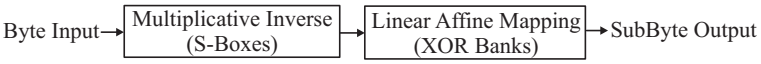


Fig. 9. Architecture of the ByteSub Submodule

Fig. 10. Architecture of the ShiftRow Submodule

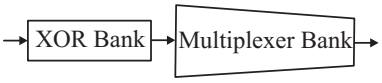


Fig. 11. Architecture of the MixColumn Submodule

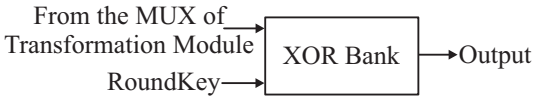


Fig. 12. Architecture of the AddRoundKey Submodule

Input" is replaced by its corresponding value from the S-box, which then goes through linear affine mapping to produce "SubByte Output", i.e. substituted byte output.

ShiftRow Submodule: The architecture shown in Fig. 10 shifts the position of bytes in the states by amounts of offsets. In this transformation, the rows of the block state are shifted over different offsets. The amount of shifts is determined by the block length. Although the first row is not shifted, the 2nd row is shifted to the left once, the 3rd row is shifted to the left twice, and the 4th is shifted thrice. Considering the offset by which a row should be shifted, the proposed architecture implements the shift row operation by using combinational logic and a parallel bank of multiplexers.

MixColumn Submodule: The elements of columns in a state are considered coefficients of a polynomial over Galois field $GF(2^8)$, where these elements are smaller than three. This polynomial is then multiplied by the fixed polynomial $[C(x) = (03)x^3 + (01)x^2 + (01)x + (02)]$ modulo $(x^4 + 1)$. The operation could be carried out by using matrix operations. The column mixing step is basically a matrix multiplication in the Galois field, which is carried out by using shift and XOR operations. The multiplexer bank of this architecture is different from ShiftRow Submodule, which is shown in Fig. 11. In this case, each XOR gate from the XOR bank is connected to each multiplexer from the multiplexer bank to perform Galois field multiplication. Then these gates are connected in an array to perform the operation for 128 bits.

AddRoundKey Submodule: The high-level architecture of the AddRoundKey submodule is shown in Fig. 12. In this case, the round key obtained from the key scheduler is XORed with the block state obtained from the MixColumn transformation or ShiftRow transformation based on the type of round being implemented. In the standard round, the round key is XORed with the output obtained from the MixColumn transformation, whereas in the final round, it is XORed with the output obtained from the ShiftRow transformation. Moreover, bitwise XOR operation is performed between the initial round key and the initial state block in the initial round.

6.2.2. *Key Scheduler Module*

The round key is obtained from the initial key through key expansion by using the architecture in Fig. 13. If it is the first round in the standard round module, the multiplexer outputs the initial key for expansion. The module is able to generate subsequent round keys from the initial round keys through the register. As a result, round keys are generated

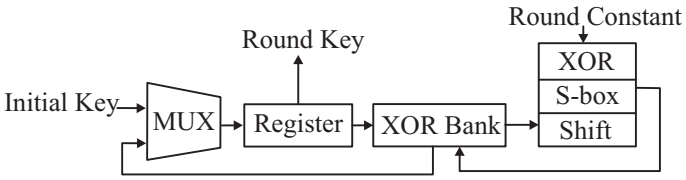


Fig. 13. Architecture of the Key Scheduler Module

at every round, thereby reducing area requirements. The ByteSub operation in the key schedule is implemented by using S-boxes. Round-key computation is completed in one clock cycle. The round constants are generated using the combinational logic [36,33]. This is amenable for hardware realization using simple XOR operations. The round constant is obtained by XORing the previous round constant by x (hexadecimal value 02). The total number of round constants generated is equal to the number of rounds.

6.2.3. *Controller module*

The sequence of operation of the system is determined by the control module. The multiplexer selects inputs, whereas the control module provides register load signals. The controller (not presented for brevity) is implemented as a finite state machine (FSM) with 13 states [33]. Each state of the FSM represent one round of operation; in addition there are two load register rounds and one initial state.

6.2.4. *Decryption in AES*

In the Rijndael algorithm, the encryption and decryption use the same operations, but in different order [36,33]. In the decryption, inverse transformations of the round functions are applied. The sequence in which the transformations of the round function are applied differs from that in the encryption. For encryption, initial AddRoundKey involves XORing of input key with the plain text. The second through tenth key addition involves XORing of the round key with the MixColumn output for encryption and the inverse of the ByteSub output for decryption. The final key addition involves XORing of the final round key with the output of the ShiftRow for encryption and the inverse of the ByteSub for decryption, respectively. The MixColumn transformation in decryption use a different fixed polynomial $D(x)$ than that of MixColumn transformation in encryption $C(x)$. The fixed polynomial for decryption is represented as: $[D(x) = (0B)x^3 + (0D)x^2 + (09)x + (0E)]$ modulo $(x^4 + 1)$. The MixColumn transformation hardware in decryption has same structure that uses XOR bank and multiplexer bank, however operating on different inputs for different polynomials [36,33].

7. Simulations, Experiments, and Field-Programmable-Gate-Array (FPGA) Prototyping

7.1. *Experiments with the Algorithm*

7.1.1. *Experimental Setup*

The proposed algorithm is implemented in MATLAB. The computation platform was a Core 2 duo processor with a speed of $2.8GHz$ and $4GB$ of memory. A larger volume of benchmark images and watermarks were used for the experiments, but results for selected images and watermarks are presented for brevity.

7.1.2. *Insertion Algorithm Testing and Watermarked Image Quality Assessment*

The defaults values for α_{AC} and α_{DC} are set to 0.1 and 0.02, respectively. These values have been found to yield optimal results. Selected binary watermarks are presented in Fig. 14. In Fig. 15 and 16, selected watermarked benchmark images are presented with the first image of Fig. 14 used as a watermark. It was observed that the typical execution time for insertion was fraction of seconds for an image with a size of 256×256. The algorithm performance is independent of image size and hence any size image can be considered. Thus, the time overhead of the algorithm was minimal. The storage requirement overhead was also very small because only the keys are needed to be stored by the owner to prove ownership. The amount of memory required to store the keys is insignificant compared to the amount required for the host image.

(a) Sample 1 (b) Sample 2

Fig. 14. Sample Binary Images Used as Watermarks

The quality of the watermarked images obtained by using our watermarking algorithm may be assessed by visual inspection of the watermarked images in Fig. 15 and 16. In addition, the peak-signal-to-noise ratio ($PSNR$) of the watermarked images are calculated for quantitative assessment of the quality. The $PSNR$ of the watermarked images is calculated using the following expression [16,26,37,38]:

$$PSNR = 20 \times log_{10}\left(\frac{255}{RMSE}\right), \tag{2}$$

where the root mean square error ($RMSE$) of the watermarked image with respect to the original image is estimated by the following expression [16,26,37,38]:

$$RMSE = \sqrt{\left(\frac{\sum_{i=1}^{ROW}\sum_{j=1}^{COL}\sum_{k=1}^{3}\left|I\left(i,j,k\right) - I^{*}\left(i,j,k\right)\right|}{3 \times ROW \times COL}\right)}, \tag{3}$$

where i is the image pixel row from 1 to ROW, j is the image pixel column from 1 to COL, and k is from 1 to 3 as RGB color pixels. $I\left(i,j,k\right)$ and $I^{*}\left(i,j,k\right)$ are the image pixels after and before watermarking, respectively. The $PSNR$ values of the sample color images are provided in Table 1.

In order to get a comparative perspective of the performance of the watermarking process, selected watermarking techniques are discussed. The watermarking algorithm proposed in this work is unique because it considers: both AC and DC DCT coefficients, uses encryption in the framework, and hides binary image inside host image by selective addition and subtraction based on watermark image gray value together in a unified approach (as discussed in preceding sections). Thus, it can allow secure storage of sensitive information in an image. In order to get a comparative perspective of performance of our CryptMark algorithm, we selected few research works [39,40,41,42] as these algorithms hide binary image inside the host image. The $PSNR$ for [39] is the range of $24.9 - 47.9dB$ with an average of $38.3dB$, which is 62.4% lower in quality than the proposed algorithm. For the algorithm in [40], the $PSNR$ is the range of $29.1 - 39.6dB$ with an average of $34.7dB$, which is 65.9% lower than the current paper. The $PSNR$ for [41] is in the range of $23.67 - 23.73dB$ with an average of $23.7dB$, which is 76.7% lower in quality than the algorithm presented in this paper. The $PSNR$ for [42] is in the range of $42.9 - 45.8dB$ (average of $44.1dB$), which is 56.7% lower than the algorithm of the current paper. In summary, the algorithm of the current paper produces better quality watermarked images with $PSNR$ improvement in the range of $56.7 - 76.7\%$ than the above discussed papers.

7.1.3. *Authentication Algorithm Testing*

The performance of the proposed algorithm with respect to attack resilience has been established by the results shown in Table 1 for the well-known StirMark attack against the algorithm. The watermarking survived all but one of the attack types included in the synthetic benchmark attack, StirMark [43]. Only the binary outcome of different attacks has been reported in the Table 1; i.e., whether the watermark extracted has survived in the sense that it is recognizable as a replica of the original watermark. As long as the extracted watermark is recognizable, the purpose is served. There is always a tradeoff between the perceptual quality of the watermarked image produced by an algorithm and the quality of the extracted watermark under noise and other degradations. Hence, after establishing with different images that the visual quality of the watermarked images is acceptable, the results are presented that help benchmarking this algorithm against the ideal algorithm that survives all the attack types in the StirMark attack. The algorithm's extraction performance is observed to be better than the counterparts proposed in [39,41].

(a) Original Lena (b) Watermarked Lena

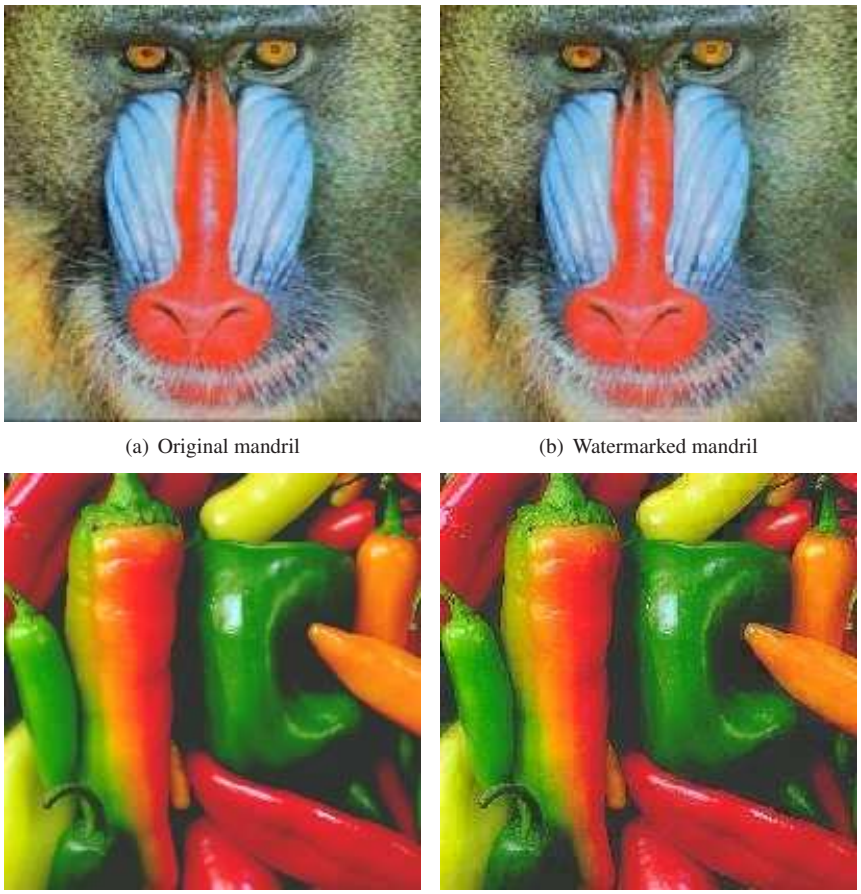
(c) Original Mig21 (d) Watermarked Mig21

(e) Original F16 (f) Watermarked F16

Fig. 15. Performance Evaluation CryptMark for a Set of Benchmark Images - Set 1

7.2. *Prototyping and Testing of the Architectures*

For the prototyping, the architecture was modeled by using VHSIC hardware description language (VHDL), and the simulation was carried out with Modelsim XE III 6.0a tools. The VHDL code was compiled with Xilinx ISE. The synthesis of the architectures was carried out by using VIRTEX-II technology. The power measurement of architectures



(a) Original mandril (b) Watermarked mandril

(c) Original pepper (d) Watermarked pepper

Fig. 16. Performance Evaluation CryptMark for a Set of Benchmark Images - Set 2

Table 1
Attacks Performed using Benchmarks for Testing of the Invisible-Robust Algorithm

Attacks Performed for Testing	Various Benchmark Images				
	Lena ($PSNR = 105$)	Mig21 ($PSNR = 97$)	F16 ($PSNR = 99$)	mandril ($PSNR = 101$)	pepper ($PSNR = 108$)
JPEG Compression 0 quality	Survived	Survived	Survived	Survived	Survived
Gray scaling 16 levels	Survived	Survived	Survived	Survived	Survived
Gray scaling 256 levels, JPEG compression 0 quality	Survived	Survived	Survived	Survived	Survived
Blurring , 0 quality JPEG Compression	Survived	Survived	Survived	Survived	Survived
Partial cropping	Survived	Survived	Survived	Survived	Survived
Stirmark Self Similarities	Survived	Survived	Survived	Survived	Survived
Stirmark 0 quality JPEG compression	Survived	Survived	Survived	Survived	Survived
Stirmark median filtering	Survived	Survived	Survived	Survived	Survived
Stirmark Random Distortions	Survived	Survived	Survived	Survived	Survived

is performed using XPower. The throughput of architectures is calculated using the following expression:

Table 2
Synthesis of the Watermarking Architecture

Architecture Attributes	Architecture Statistics
Cell Usage – Basis Logic Gates, LUTs, MUX, XOR	218 $BELS$
Device Usage – Number of Slices	78
Maximum Frequency	256 MHz
Minimum Period	3.9 ns
Critical Path Delay	2.1 ns
Power Dissipation	3.7 mW
Throughput	544.2 $Mbps$

Table 3
Comparative Perspective with Existing Image Watermarking Architectures

Prior Research	Architecture Features
Jeong, et al. [44]	Invisible-Robust, Video, Wavelet, XILINX VERTEX2PRO
Jeong, et al. [45]	Invisible-Robust, Video, Spatial, ALTERA STRATIX FPGA
Mohanty, et al. [2]	Invisible-Robust, Image, Spatial, $50MHz$
Seo, et al. [46]	Invisible-Robust, Image, Wavelet, $82MHz$
Petitjean, et al. [47]	Invisible-Robust, Image, Fractal, $50MHz$
Maes, et al. [48]	Invisible-Robust, Video, Spatial, $17/14\ kG$ Logic
This Paper	Invisible-Robust, Image, DCT, $256MHz$

$$\text{Architecture Throughput} = \left(\frac{\text{Quantity of Data Processed} \times \text{Clock Cycle Frequency}}{\text{Total Number of Clock Cycles}}\right), \tag{4}$$

$$= \left(\frac{\text{Quantity of Data Processed}}{\text{Critical Path Delay} \times \text{Total Number of Clock Cycles}}\right). \tag{5}$$

7.2.1. *The Watermarking Architecture*

First, the architecture was tested for functionality. It resulted in same outputs as that expected from the algorithms, thus proving its correctness. Synthesis results for the high-performance invisible-robust watermarking unit of the SDC are presented in Table 2. A pixel i.e. 8 bit data is processed through 7 cycles each taking critical-path-delay time of $2.1ns$; thus the throughput is $8/(2.1\ ns \times 7)$ or $544.2\ Mbps$.

The comparative perspective of this architecture with respect to existing FPGA-based image watermarking schemes is presented in Table 3. It may be noted that the working domain and scope of these schemes are different; so a fair comparison is not possible. However, from a larger perspective, the architecture proposed in this paper has minimal cell usage and higher performance compared to the existing works.

7.2.2. *The Encryption Architecture*

The synthesis results of the FPGA prototyping of the encryption unit of the SDC is presented in Table 4. A pixel i.e. 8 bit data is processed through 12 cycles each taking critical-path-delay time of $4.3ns$; thus the throughput is $8/(4.3\ ns \times 12)$ or $155.0\ Mbps$. However, the AES architecture can take 128 bit data at a time and in that case the throughput is $128/(4.3\ ns \times 12)$ or $2480.6\ Mbps$ or $2.480\ Gbps$. We can utilize this throughput by processing 16 pixels at a time. The implementation of the proposed architecture is presented in Table 5 in the context of similar works to get a comparative perspective. However, it is difficult to make a fair comparison as various works use different architectures, design objectives, and hardware resources.

It may be noted that the frequency reported for watermarking and encryption in Table 2 and 4, respectively are "maximum" possible operating for these architectures. However, when they are integrated in the SDC, they may operate in two different scenario. If the whole SDC system is operated using a centralized controller in single operating

Table 4
Synthesis of the Encryption Architecture

Architecture Attributes	Architecture Statistics
Cell Usage – Basis Logic Gates, LUTs, MUX, XOR	6117 $BELS$
Device Usage – Number of Slices	968
Maximum Frequency	204 MHz
Minimum Period	4.9 ns
Critical Path Delay	4.3 ns
Power Dissipation	39.8 mW
Throughput (Processing a pixel at a time)	155.0 $Mbps$
Throughput (Processing a AES 128-bit data packet at a time)	2.48 $Gbps$

Table 5
Comparative Perspective with Existing Encryption Architectures

Prior Research	Architecture Features
Laue, et. al. [49]	Compact AES module design, 125.6 MHz, 916 slices, 171.1 $Mbps$
Badillo, et. al. [50]	96.42 MHz, 586 slices, 1.45 $Gbps$
Sodon, et. al. [51]	Low-cost using bit-serial approach, 510 MHz maximum frequency, 0.37 $Gbps$
McLoone, et. al. [52]	Virtex-E FPGA, 192-bit key design at 5.8 $Gbps$
Chittu, et al. [53]	Xilinx FPGA, 75 MHz, 739 $Mbps$
Sklavos, et al. [54]	XCV300-BG432, 2358 Slices, 22 MHz, 259 $Mbps$
This Paper	Single-Pixel Processing at 155.0 $Mbps$, 16-Pixel Processing at 2.48 $Gbps$

frequency mode (e.g. [16]), then that frequency is the minimal operating frequency of the modules. On the other hand, if they are operated in decentralized controlled fashion with multiple operating frequency (e.g. [34]), then they units can operate at any frequency up to their maximum operating frequency.

8. Conclusions and Future Research

8.1. *Summary and Conclusions*

This paper introduced the concept of an SDC for image copyright protection, security, and authentication in a DRM framework. The novel invisible watermarking method called CryptMark that uses cryptography and watermarking methods simultaneously to provide double-layer protection to the digital media is presented that can be an effective technique for DRM. Exhaustive testing of the algorithm proved that the algorithm works well and can survive various forms of attacks. The architecture and the FPGA implementation of the invisible watermarking and encryption unit were presented toward the realization of the SDC. The performance of the architectures was found to be promising for real-time DRM. Design and implementation of the remaining components of the SDC are being conducted as ongoing research in our laboratory.

8.2. *Future Research*

Currently, research is under way to consider having two watermarks as a user-specific binary watermark and synthetic watermark generated by the system and fuse them into the cover for additional protection and better image quality. Other possible extensions will include the use of wavelet transforms for embedding strong watermarks. Blind extraction of invisible watermarks is also a planned extension, particularly because of its usefulness in authentication at the receiver end as well as identification of secretive communication. *W*atermarking, *En*cryption, and *Compression*

will be integrated as a single unified operation to be called *WEnCompression*, exploiting common operations in the algorithms and resources in their architectures. Further low-power methodologies for complete SDC system are in the pipeline that will also improve the portability of the SDC by enhancing battery life.

9. Acknowledgments

The author would like to acknowledge the help of many colleagues, such as M. Varanasi and O. B. Adamo of the University of North Texas (UNT).

References

[1] S. Emmanuel, M. S. Kankanhalli, A Digital Rights Management Scheme for Broadcast Video, ACM-Springer Verlag Multimedia Systems Journal 8 (6) (2003) 444–458.
[2] S. P. Mohanty, R. K. C., S. Nayak, FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder, in: Lecture Notes in Computer Science, Vol. 3356, 2004, pp. 344–353.
[3] I.J.Cox, J. Kilian, T. Leighton, T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, IEEE Transactions on Image Processing 6 (12) (1997) 1673–1687.
[4] S. Craver, N. Memon, B. L. Yeo, M. M. Yeung, Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitations, Attacks and Implications, IEEE Journal on Selected Areas in Communications 16 (4) (1998) 573–586.
[5] H. Guo, N. D. Georganas, A Novel Approach to Digital Image Watermarking Based on a Generalized Secret Sharing Scheme, ACM-Springer Verlag Multimedia Systems Journal 9 (3) (2003) 228–238.
[6] G. Jiang, M. Yu, S. Shi, X. Liu, Y. D. Kim, New Blind Image Watermarking in DCT Domain, in: Proceedings of the 6th International Conference on Signal Processing, Vol. 2, 2002, pp. 1580 – 1583.
[7] Z. M. Lu, D. G. Xu, S. H. Sun, Multipurpose Image Watermarking Algorithm based on Multistage Vector Quantization, IEEE Transactions on Image Processing 14 (6) (2005) 822–831.
[8] Y. T. Pai, S. J. Ruan, J. Götze, Energy-Efficient Watermark Algorithm Based on Pairing Mechanism, in: Lecture Notes in Computer Science (LNCS), KES (1), 2005, pp. 1219–1225.
[9] N. P. Sheppard, R. S. Naini, P. Ogunbona, On Multiple Watermarking, in: Proceedings of the ACM Multimedia workshops on multimedia and security: new challenges, 2001, pp. 3–6.
[10] N. Memon, P. W. Wong, Protecting Digital Media Content, Communications of the ACM 41 (7) (1998) 34–43.
[11] M. Swanson, M. Kobayashi, A. Tewfik, Multimedia Data Embedding and Watermarking Technologies, Proceedings of the IEEE 86 (6) (1998) 1064–1087.
[12] H. Berghel, Watermarking Cyberspace, Communications of the ACM 40 (11) (1997) 19–24.
[13] S. P. Mohanty, N. Memon, K. Chatha, Guest editorial, Special Issue on Circuits and Systems for Real-Time Security and Copyright Protection of Multimedia, Elsevier International Journal on Computers and Electrical Engineering (IJCEE) 35 (2) (2009) 231–234.
[14] S. P. Mohanty, R. Sheth, A. Pinto, M. Chandy, CryptMark: A Novel Secure Invisible Watermarking Technique for Color Images, in: Proceedings of the 11th IEEE International Symposium on Consumer Electronics (ISCE), 2007, pp. 1–6.
[15] Federal information processing standards publication 197 (fips 197), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.
[16] S. P. Mohanty, N. Rangnathan, R. K. Namballa, A VLSI Architecture for Visible Watermarking in a Secure Still Digital Camera (S^2DC) Design, IEEE Transactions on Very Large Scale Integration Systems 13 (7) (2005) 808–818.
[17] S. P. Mohanty, O. B. Adamo, E. Kougianos, Vlsi architecture of an invisible watermarking unit for a biometric-based security system in a digital camera, in: Proceedings of the 25th IEEE International Conference on Consumer Electronics (ICCE), 2007, pp. 485–486.
[18] J. L. Trpanier, M. Sawan, Y. Audet, J. Coulombe, A wide dynamic range cmos digital pixel sensor, in: Proceedings of the International Symposium on Circuits and Systems, 2002, pp. 437–440.
[19] O. Y. Pecht, G. R. Nelson, G. A. Jullien, Patent title: Digital watermarking cmos sensor, patent application number: 20070019090, Patent, http://www.freshpatents.com/Digital-watermarking-cmos-sensor-dt20070125ptan20070019090.php (May 19 2005).
[20] E. Artyomov, O. Y. Pecht, Adaptive multiple resolution cmos active pixel sensor, IEEE Transactions on Circuits and Systems 53 (10) (2006) 21782186.
[21] S. Ravi, A. Raghunathan, P. Kocher, S. Hattangady, Security in embedded systems: design challenges, ACM Transactions on Embedded Computing Systems 3 (3) (2004) 461491.
[22] G. L. Friedman, The Trustworthy Digital Camera: Restoring Credibility to the Photographic Image, IEEE Transactions on Consumer Electronics 39 (4) (1993) 905–910.
[23] P. Blythe, J. Fridrich, Secure digital camera, in: Proceedings of Digital Forensic Research Workshop (DFRWS), 2004.
[24] S. P. Mohanty, N. Rangnathan, R. K. Namballa, VLSI Implementation of Visible Watermarking for a Secure Digital Still Camera Design, in: Proceedings of the 17th International Conference on VLSI Design, 2004, pp. 1063–1068.

[25] L. D. Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, G. Depovere, Implementation of a Real-Time Digital Watermarking Process for Broadcast Monitoring on Trimedia VLIW Processor, IEE Proceedings on Vision, Image and Signal Processing 147 (4) (2000) 371–376.
[26] N. J. Mathai, D. Kundur, A. Sheikholeslami, Hardware Implementation Perspectives of Digital Video Watermarking Algortithms, IEEE Transanctions on Signal Processing 51 (4) (2003) 925–938.
[27] T. H. Tsai, C. Y. Lu, A Systems Level Design for Embedded Watermark Technique using DSC Systems, in: Proceedings of the IEEE International Workshop on Intelligent Signal Processing and Communication Systems, 2001.
[28] A. Garimella, M. V. V. Satyanarayan, R. S. Kumar, P. S. Murugesh, U. C. Niranjan, VLSI Impementation of Online Digital Watermarking Techniques with Difference Encoding for the 8-bit Gray Scale Images, in: Proceedings of the International Conference on VLSI Design, 2003, pp. 283–288.
[29] Y. C. Fan, L. D. Van, C. M. Huang, H. W. Tsao, Hardware-efficient architecture design of wavelet-based adaptive visible watermarking, in: Proceedings of 9th IEEE International Symposium on Consumer Electronics, 2005, p. 399403.
[30] G. R. Nelson, G. A. Jullien, O. Y. Pecht, Cmos image sensor with watermarking capabilities, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2005, p. 53265329.
[31] L. Tian, H.-M. Tai, Secure images captured by digital camera, in: Proceedings of the International Conference on Consumer Electronics, 2006, pp. 341–342.
[32] R. Lukac, K. N. Plataniotis, Secure single-sensor digital camera, IEE Electronics Letters 42 (11) (2006) 627–629.
[33] O. B. Adamo, S. P. Mohanty, E. Kougianos, M. Varanasi, VLSI Architecture for Encryption and Watermarking Units Towards the Making of a Secure Digital Camera, in: Proceedings of the IEEE International SOC Conference (SOCC), 2006, pp. 141–144.
[34] S. P. Mohanty, N. Ranganathan, K. Balakrishnan, A dual voltage-frequency vlsi chip for image watermarking in dct domain, IEEE Transaction on Circuits & Systems II 53 (5) (2006) 394–398.
[35] N. H. E. Weste, D. Harris, Principles of CMOS VLSI Design: A Systems Perspective, 3rd Edition, Addison Wesley Longman, 2006.
[36] N. M. Kosaraju, M. Varanasi, S. P. Mohanty, A high-performance vlsi architecture for advanced encryption standard (aes) algorithm, in: 19th International Conference on VLSI Design, 2006, pp. 481–484.
[37] I. E. G. Richardson, H.264 and MPEG-4 Video Compression, John Wiley & Sons, Ltd., 2003.
[38] J. Chen, U. V. Koc, K. J. R. Liu, Design of Digital Video Coding Systems A Complete Compressed Domain Approach, Marcel Dekker, Inc., 2002.
[39] Y. Wu, X. Guan, M. S. Kankanhalli, Robust invisible watermarking of volume data using the 3d dct, in: Proc. of Computer Graphics International (CGI), 2001, pp. 359–362.
[40] Y. T. Pai, S. J. Ruan, Low Power Block-Based Watermarking Algorithm, IEICE Transactions on Information and Systems E89-D (4) (2006) 1507–1514.
[41] V. Saxena, J. P. Gupta, Collusion attack resistant watermarking scheme for images using dct, in: Proceedings of 15th IEEE Conf. on Signal Processing and Communications Applications, 2007, pp. 1–4.
[42] N. N. Rao, P. Thrimurthy, B. R. Babu, A Novel Scheme for Digital Rights Management of Images Using Biometrics, International Journal of Computer Science and Network Security 9 (3) (2009) 157–167.
[43] S. Voloshynovskiy, S. Pereira, T. Pun, J. Eggers, J. Su, Attacks on Digital Watermarks: Classification, Estimation-based Attacks and Benchmarks, IEEE Communications Magazine 39 (9) (2001) 118–126.
[44] Y.-J. Jeong, K.-S. Moon, J.-N. Kim, Implementation of Real Time Video Watermark Embedder Based on Haar Wavelet Transform Using FPGA, in: Proceedings of the Second International Conference on Future Generation Communication and Networking Symposia (FGCNS), 2008, pp. 63 – 66.
[45] Y.-J. Jeong, W.-H. Kim, K.-S. Moon, J.-N. Kim, Implementation of Watermark Detection System for Hardware Based Video Watermark Embedder, in: Proceedings of the Third International Conference on Convergence and Hybrid Information Technology (ICCIT), 2008, pp. 450 – 453.
[46] Y. H. Seo, D. W. Kim, Real-Time Blind Watermarking Algorithm and its Hardware Implementation for Motion JPEG2000 Image Codec, in: Proceedings of the 1st Workshop on Embedded Systems for Real-Time Multimedia, 2003, pp. 88–93.
[47] G. Petitjean, J. L. Dugelay, S. Gabriele, C. Rey, J. Nicolai, Towards Real-time Video Watermarking for Systems-On-Chip, in: Proceedings of the IEEE International Conference on Multimedia and Expo (Vol. 1), 2002, pp. 597–600.
[48] M. Maes, T. Kalker, J. P. M. G. Linnartz, J. Talstra, G. F. G. Depovere, J. Haitsma, Digital Watamarking for DVD Video Copyright Protection, IEEE Signal Processing Magazine 17 (5) (2000) 47–57.
[49] R. Laue, O. Kelm, S. Schipp, A. Shoufan, S. A. Huss, Compact aes-based architecture for symmetric encryption, hash function, and random number generation, in: Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), 2007, pp. 480–484.
[50] I. Algredo-Badillo, C. F. Uribe, R. Cumplido, Design and implementation of an fpga-based $1.452gbps$ non-pipelined aes architecture, in: Proceedings of the International Conference Computational Science and Its Applications (ICCSA), 2006, pp. 456–465.
[51] O. J. Hernandez, T. Sodon, M. Adel, Low-Cost Advanced Encryption Standard (AES) VLSI Architecture: A Minimalist Bit-Serial Approach, in: Proceedings of the IEEE Southeast Conference, 2005, pp. 121–125.
[52] M. McLoone, J. V. McCanny, Rijndael FPGA Implementations Utilizing Look-Up Tables, Journal of VLSI Signal Processing Systems, KAP 34 (3).
[53] C. Chitu, D. Chien, C. Chien, I. Verbauwhede, F. Chang, A Hardware Implementation in FPGA of the Rijndael Algorithm, in: Proceedings of the 45th Midwest Symposium on Circuits and Systems, 2002, pp. 507–510.
[54] N. Sklavos, O. Koufopavlou, Architectures and vlsi implementations of the aes-proposal rijndael, IEEE Transactions on Computers 51 (12) (2002) 1454–1459.