Journal of Real-Time Image Processing (2024) 21:83
https://doi.org/10.1007/s11554-024-01450-8

RESEARCH

Software and hardware realizations for different designs of chaos-based secret image sharing systems

Bishoy K. Sharobim[1,4] · Muhammad Hosam[1] · Salwa K. Abd-El-Hafiz[2] · Wafaa S. Sayed[2] · Lobna A. Said[1] · Ahmed G. Radwan[2,3]

Received: 14 December 2023 / Accepted: 1 March 2024

Abstract
Secret image sharing (SIS) conveys a secret image to mutually suspicious receivers by sending meaningless shares to the participants, and all shares must be present to recover the secret. This paper proposes and compares three systems for secret sharing, where a visual cryptography system is designed with a fast recovery scheme as the backbone for all systems. Then, an SIS system is introduced for sharing any type of image, where it improves security using the Lorenz chaotic system as the source of randomness and the generalized Arnold transform as a permutation module. The second SIS system further enhances security and robustness by utilizing SHA-256 and RSA cryptosystem. The presented architectures are implemented on a field programmable gate array (FPGA) to enhance computational efficiency and facilitate real-time processing. Detailed experimental results and comparisons between the software and hardware realizations are presented. Security analysis and comparisons with related literature are also introduced with good results, including statistical tests, differential attack measures, robustness tests against noise and crop attacks, key sensitivity tests, and performance analysis.

Keywords Chaos · FPGA · Secret image sharing · SHA-256 · Visual secret sharing

1 Introduction

Digital data have become essential to modern telecommunications, especially where vast images are stored and transferred. This increased the awareness of privacy and information security, and made protecting digital images a very important requirement. As a result, research efforts increased in the information security fields such as cryptography, information hiding, and secret sharing (SS) [1].

SS is a relatively new idea introduced by Shamir in 1979, where a secret number is sent to a group of participants as n shares of the secret in a meaningless form [2]. Each share alone does not give any information about the secret number, while a group of k or more shares can reveal the secret, where $k \leq n$. The idea was based on polynomial interpolation, and it is useful when the recipients are mutually suspicious or must cooperate. It is also used in cloud computing and distributed storage [1].

The idea of SS was improved to work for images in 1995 by Naor and Shamir, who introduced Visual Secret Sharing (VSS) [3]. In VSS, the recovery process is as easy as stacking the shares to recover the secret image using the human visual system. Stacking images is equivalent to the boolean OR operation between the images [4]. More secure systems were needed, which led to the introduction of Secret Image Sharing (SIS) by Thien and Lin in 2002 [5]. They used polynomial interpolation with shares of size $1/k$ of the secret image, but it needed more computation power compared to VSS.

This work is supported by the Science, Technology, and Innovation Funding Authority (STIFA), Egypt, under grant number 45631.

✉ Bishoy K. Sharobim
b.kamal2160@nu.edu.eg

1 Nanoelectronics Integrated Systems Center (NISC), Nile University, Giza 12588, Egypt

2 Engineering Mathematics and Physics Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt

3 School of Engineering and Applied Sciences, Nile University, Giza 12588, Egypt

4 Centre of Informatics Science, School of Information Technology and Computer Science, Nile University, Giza, 12588, Egypt

The need for acceleration and easily integrating encryption into existing systems led to the use of field programmable gate arrays (FPGAs) as pivotal tools in the realms of both cryptographic operations and VSS. Their distinctive ability to be customized for specific tasks, coupled with their prowess in parallel processing, has propelled them to the forefront of secure data processing [6]. Security applications often favor FPGAs over general-purpose computers because of their low power consumption, high throughput, design adaptability, cost-effectiveness in development per unit, rapid processing speed, resilience to noise, and elevated security levels [7, 8].

This work presents a VSS system as a main block for SIS to ensure fast recovery. Then, two new lossless (n, n)-SIS systems are introduced for sharing binary, grayscale, or color images using the VSS system as the backbone. The first SIS system uses the Lorenz chaotic system as a source of randomness, utilizes the generalized Arnold transform to perform permutations, and has a long and sensitive system key. The second SIS system further enhances security and robustness using SHA-256 and RSA public-key cryptosystem. Software implementations and FPGA realizations, including all the used modules, are presented for the three systems. Security analysis is performed between the secret image and shares, and validated hardware results are presented. The experimental results show the systems' effectiveness when deployed on FPGAs, exhibiting real-time processing capabilities and minimal resource utilization. Performance analysis and comparisons with recent approaches are also presented. The results demonstrate that the proposed enhanced system is a secure, robust and efficient SIS system.

The next section of this paper briefly reviews the recent related approaches of secret sharing. Section 3 describes the background needed for the proposed systems. Section 4 describes the VSS system, and Sect. 5 describes how the VSS system is modified to create the first SIS system. Section 6 describes the second SIS system. Section 7 describes the hardware implementations for the three systems. Section 8 gives the results and comparisons, and Sect. 9 briefly gives the conclusions and future work.

2 Related work

Most VSS systems use halftoning to convert all types of images into binary images and process them. Halftoning represents the image as dots, which affects the quality of the images [9]. Due to data loss when OR is used in recovery, XOR is used in recent literature to provide better quality for the recovered images [10]. There are different types of VSS introduced for different purposes, such as weighted VSS, which gives different weights for shares, and the total weight available in the recovery process defines the quality of the recovered image [11]. Another type is the tagged shares, which adds information in each share to differentiate between shares by folding the share, for example, to show the tag [12]. Others added features like meaningful shares [13] or sharing multiple images [14].

As previuosly mentioned, the recovered image quality in VSS increases when using XOR. Hence, it is the primary recovery method for recently proposed systems. A lossless system was introduced for binary and grayscale images by converting the image into a bitstream and sharing it using pixel vectorization [15]. Another proposed system prioritized different shares, giving high importance to some shares that can be used in recovery and give better quality, but this system was lossy [16]. Another system requires some essential shares to be present in the recovery process to recover the secret, and it can not be recovered without including those essential shares [17]. While VSS is simple, it is not robust to noise and lacks good quality for the recovered images in most systems.

On the other hand, the common techniques used in SIS systems are the Chinese Remainder Theorem (CRT) and polynomial interpolation, where the size of the shares can be a portion of the secret size or with fixed size [18]. Some features are also added to SIS, such as meaningful shares [19] and sharing multiple secrets [20]. A system for grayscale images was proposed that produced fixed size shares of 23×23 [18]. Other systems used CRT to have smaller shares with size $1/k$ of the secret image size [21]. Another system prioritized participants, where high priority shares recover the secret with higher quality [22]. Quick Response (QR) codes are also used to conceal shares and make them less suspicious[23]. Another system used CRT, XOR and a modification of Shamir's secret sharing to share different types of images [24].

3 Background

This section describes the background needed for the proposed systems, including the Lorenz chaotic system which is used as a source of randomness, the generalized Arnold transform which is a permutation algorithm, SHA-256 which is a hash function, and the evaluation criteria used to test the proposed systems.

3.1 The Lorenz chaotic system

The Lorenz system is used as a Pseudo Random Number Generator (PRNG) of the proposed systems, and it has the following equation [25]:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z), \quad \frac{dz}{dt} = xy - \beta z, \tag{1}$$

where σ, ρ, and β are the system's parameters. The system is solved by Euler method using the following formulae [26]:

256 bits

	k_1	k_2	k_3	k_4	k_5	k_6
Key Part	43 bits	43 bits	43 bits	43 bits	42 bits	42 bits
Equation	$x_0 = \frac{k_1}{2^{41}} + 2^{-10}$	$y_0 = \frac{k_2}{2^{41}} + 2^{-10}$	$z_0 = \frac{k_3}{2^{41}} + 2^{-10}$	$\sigma = \frac{k_4}{2^{42}} + 8$	$\beta = \frac{k_5}{2^{41}} + 2$	$\rho = \frac{k_6}{2^{41}} + 32$
Range	$[2^{-10}, 4 + 2^{-10}[$	$[2^{-10}, 4 + 2^{-10}[$	$[2^{-10}, 4 + 2^{-10}[$	$[8, 10[$	$[2, 4[$	$[32, 34[$

Fig. 1 Subkeys generation

$$x_{t+1} = x_t + h(\sigma(y_t - x_t)), \tag{2a}$$

$$y_{t+1} = y_t + h(x_t(\rho - z_t)), \tag{2b}$$

$$z_{t+1} = z_t + h(x_t y_t - \beta z_t), \tag{2c}$$

where $h = 2^{-7}$ is the time step, and x_0, y_0 and z_0 are the initial conditions.

The step h is chosen as a negative power of 2 to be used in the hardware implementation as shifting instead of multiplication [27].

The following equation is used to extract l bits from x:

$$\lfloor x_t \times sf \rfloor \mod 2^l, \tag{3}$$

where sf is a scaling factor chosen to ensure randomness, $\lfloor \cdot \rfloor$ is the floor function, and similar equations are used for y and z.

To generate the initial conditions and parameters for the Lorenz system, a system key is used with a size of 256 bits to resist brute force attacks. Figure 1 illustrates how to extract the initial conditions and parameters, with the ranges required for the chaotic operation of the Lorenz system.

The same extraction process is performed in the generation and recovery schemes, using the same system key, to guarantee the correct operation of both sides of the sharing system. The randomness test results for the Lorenz chaotic system are provided in the supplementary file.

3.2 Generalized Arnold transform

Arnold transform is used to permute the pixels of an image, where a pixel in the position (x, y) is transformed to the new position (x', y') [28]. The generalized Arnold transform has the following equation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \gamma \\ \zeta & \gamma\zeta + 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mod N, \tag{4}$$

where γ and ζ are the parameters for the transformation, and N is the dimension of a square image. The inverse transformation is given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \gamma\zeta + 1 & -\gamma \\ -\zeta & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \mod N, \tag{5}$$

3.3 Secure hash algorithm: SHA-256

The SHA-256 is a well-known hash function that takes an input of size $< 2^{64}$ bits and maps it into a 256-bit digest [29]. The SHA-256 has a word size of 32-bit, where all the operations are performed on words. First, the functions used in the algorithm are given as follows:

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z), \tag{6a}$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), \tag{6b}$$

$$\Omega_0(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x), \tag{6c}$$

$$\Omega_1(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x), \tag{6d}$$

$$\omega_0(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x), \tag{6e}$$

$$\omega_1(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x), \tag{6f}$$

where $ROTR^n$ is circular right shift n positions and SHR^n is right shift n positions.

The SHA-256 starts with initializing two types of constants [29], the first type is a set of 64 constant words, $K_0, K_1, \cdots, K_{63}$. They come from the first 64 prime numbers, where 32 bits of the fractional parts of cube roots are used. The second type of constants is the set of initial hash values, $H_0, H_1, \cdots, H_7$, which consists of 8 words obtained from the square root of the first 8 prime numbers, where the first 32 bits of the fractional part are used.

After initializing the constants, two steps of preprocessing are done on the input. The first step is padding the message to be able to parse it into equal 512-bit blocks. For a message of size l, one bit "1" is appended to the end of the message followed by k zero bits, where k is the smallest positive solution for the equation:

$$l + 1 + k \equiv 448 \mod 512. \tag{7}$$

Then, append the binary value of l in 64 bits, to have a padded message whose size is a multiple of 512. Afterward, the padded message is parsed into 512-bit blocks. Each message block, M, consists of 16 words, $M_0, M_1, \cdots, M_{15}$, and is processed as follows, where the operations are performed on the blocks consecutively to get the final hash and all additions are mod 2^{32}:

1. Prepare W:

$$W_t = \begin{cases} M_t, & 0 \le t \le 15 \\ \omega_1(W_{t-2}) + W_{t-7} + & 16 \le t \le 63 \\ \omega_0(W_{t-15}) + W_{t-16}, & \end{cases} \tag{8}$$

2. Initialize variables for current hash:

$$a = H_0, \quad b = H_1, \quad c = H_2, \quad d = H_3, \tag{9a}$$

$$e = H_4, \quad f = H_5, \quad g = H_6, \quad h = H_7. \tag{9b}$$

3. For each word in W, $t = 0$ to 63:

$$T_1 = h + \Omega_1(e) + Ch(e,f,g) + K_t + W_t, \tag{10a}$$

$$T_2 = \Omega_0(a) + Maj(a,b,c), \tag{10b}$$

$$h = g, \quad g = f, \quad f = e, \quad e = d + T_1 \tag{10c}$$

$$d = c, \quad c = b, \quad b = a, \quad a = T_1 + T_2. \tag{10d}$$

4. Compute the new hash values after the block:

$$H_0 = a + H_0, \quad H_1 = b + H_1, \quad H_2 = c + H_2, \tag{11a}$$

$$H_3 = d + H_3, \quad H_4 = e + H_4, \quad H_5 = f + H_5, \tag{11b}$$

$$H_6 = g + H_6, \quad H_7 = h + H_7. \tag{11c}$$

After repeating the above steps for all the blocks, the final digest is formed by concatenating the final binary hash values using:

$$H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4 \,||\, H_5 \,||\, H_6 \,||\, H_7, \tag{12}$$

where || is the concatenation operation.

3.4 Evaluation criteria

The statistical security criteria and robustness of the proposed systems against different attacks are evaluated, where the tests and the attacks are described in Table 1 with the description, formula, ranges, and optimal values.

4 The VSS system

The VSS generation scheme is shown in Fig. 2a, generating n shares from the input image using the Lorenz chaotic system. First, a random number $Rand$ in the range $[1, \ n]$ is generated from Lorenz using:

$$Rand = (\lfloor x_t \times sf \rfloor \mod n) + 1, \tag{13}$$

where $sf = 2^{40}$.

Then, for each pixel P_S in the secret image I, $(n-1)$ random pixels are generated from the Lorenz chaotic system.

Each random pixel P_i is generated using:

$$P_i\{R,G,B\} = \lfloor \{x_t, y_t, z_t\} \times sf \rfloor \mod 2^8, \tag{14}$$

where the three outputs of the Lorenz chaotic system are first scaled by sf to access the fractional parts. Then, the 8 least significant bits (LSBs) are extracted using the *mod* operation to obtain three 8-bit random numbers that represent the R, G, and B channels of a random pixel, P_i.

Using P_S and the generated random pixels P_i, the n^{th} pixel P_{Rand} is calculated as

$$P_{Rand} = P_S \oplus \bigoplus_{i=1}^{n-1} P_i, \tag{15}$$

where $\bigoplus_i^n x_i$ represents an XOR of many terms.

The $(n-1)$ random pixels and P_{Rand} are distributed into the n shares according to the value of $Rand$ using the distributing table shown in Fig. 2a, where P_{Rand} is assigned to the share S_{Rand}, and the random pixels P_i are consecutively placed in the other shares. The recovery scheme is simple, as shown in Fig. 2b, where the shares are XORed to recover the secret.

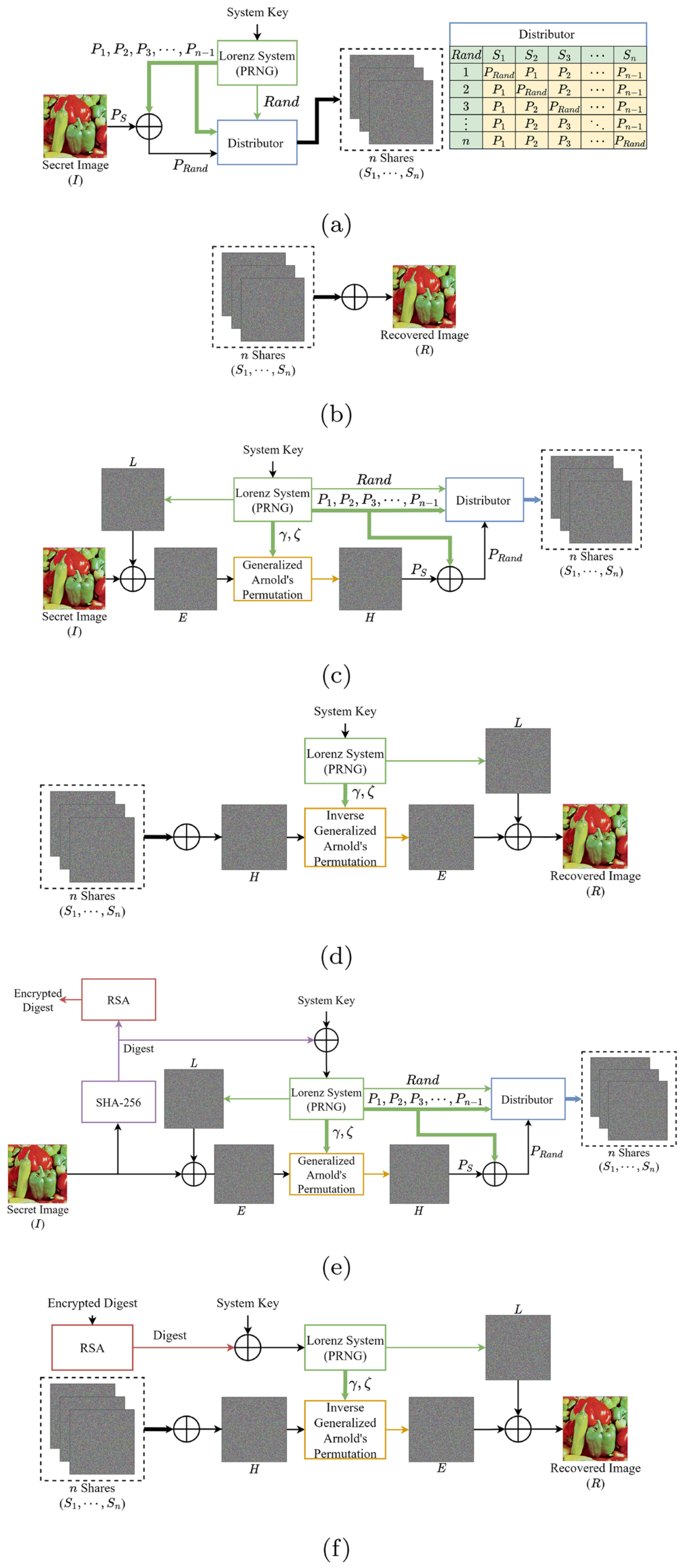
The statistical analysis results are given in [31], where the system passes all statistical tests. The robustness analysis results are shown in Table 2, where the system does not pass the crop and differential attack tests. Key sensitivity is not applicable, because no key is used in decryption. Modifications to the system will be introduced in the following sections to pass the aforementioned tests.

5 The first SIS system

The generation scheme of the first proposed SIS system, SIS-I, is shown in Fig. 2c, where it consists of substitution and permutation phases followed by the VSS system. The substitution and permutation phases enhance the system's security by incorporating the confusion and diffusion properties as defined by Shannon [32]. The permutation stage also enables the system to resist crop attacks.

Table 1 Evaluation criteria

Name	Description	Formula	Range	Optimal
Histogram	Shows the distribution of pixel values	–	Any dist	Uniform dist
Entropy	Measures the randomness in an image	$-\sum_{i=0}^{255} P(i)log_2 P(i)$, where $P(i)$ is the probability of the pixel value i	[0, 8]	8
Root Mean Square Error (RMSE)	Measures the difference between two images	$\sqrt{\frac{1}{W\times H}\sum_{i=1}^{H}\sum_{j=1}^{W}(x(i,j)-y(i,j))^2}$, where W and H are the width and height of the images, respectively, and $x(i,j)$ is the pixel value at row i and column j in the image x	[0, 255]	–
Correlation coefficient / Adjacent pixels correlation	The correlation coefficient, ρ, measures the degree of similarity between two vectors / between adjacent pixels in horizontal, vertical, and diagonal directions	$Cov(x,y)=\frac{1}{n}\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=j}^{n}x_j)(y_i-\frac{1}{n}\sum_{i=j}^{n}y_j)$, $D(x)=\frac{1}{n}\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=j}^{n}x_j)^2$, $\rho=\frac{Cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}$,	$[-1, 1]$	0
NIST SP 800-22	A group of 15 tests of randomness issued by NIST [30]. The P-value is calculated for each test, with a selection of an $\alpha = 0.01$ or 0.001 for encryption applications"	[0, 1]	[0, 1]	P-value ≥α
Noise attacks	Noise is added to the shares, and the recovered image is investigated to study the effect of noise by measuring the Peak signal-to-Noise Ratio (PSNR)	$\text{PSNR} = 20\log_{10}\frac{255}{RMSE}$	$[0, \infty]$	–
Crop attack	Parts of the shares are cropped, and the recovered image is investigated to study the effect of cropped parts	PSNR	$[0, \infty]$	–
Key sensitivity	One bit is changed in the key during decryption and the RMSE and Bit Error Rate (BER) are measured between the original image and the wrongly recovered image	RMSE, BER	RMSE [0, 255] BER [0, 1]	BER 0.5
Differential attacks	Study the impact of changes in the plain image on the cipher image. A change in the LSB of a random pixel in the original image is done. The Number of Pixel Change Rate (NPCR) and the Unified Average Changing Intensity (UACI) are measured between the cipher images from the original image E_1 and the the modified image E_2	$D(i,j)=\begin{cases}0 & \text{if } (E_1(i,j)=E_2(i,j)),\\ 1 & \text{if } (E_1(i,j)\neq E_2(i,j))\end{cases}$ $\text{NPCR}=\frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W}D(i,j)\times 100\%$ $\text{UACI}=\frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\frac{\lvert E_1(i,j)-E_2(i,j)\rvert}{255}\times 100\%$	[0, 100]	NPCR 99.61% UACI33.46%

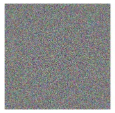
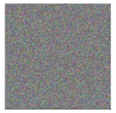
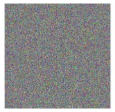
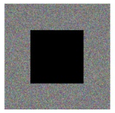
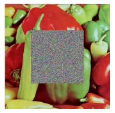
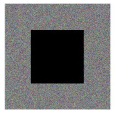
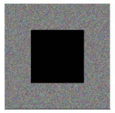


Distributor

$Rand$	S_1	S_2	S_3	$\cdots$	S_n
1	P_{Rand}	P_1	P_2	$\cdots$	P_{n-1}
2	P_1	P_{Rand}	P_2	$\cdots$	P_{n-1}
3	P_1	P_2	P_{Rand}	$\cdots$	P_{n-1}
$\vdots$	P_1	P_2	P_3	$\ddots$	P_{n-1}
n	P_1	P_2	P_3	$\cdots$	P_{Rand}

Fig. 2 Block diagrams of the proposed (a) VSS generation, (b) VSS recovery, (c) SIS-I generation, (d) SIS-I recovery, (e) SIS-II generation, and (f) SIS-II recovery

Table 2 Results of different attacks

Criteria	**VSS**	**SIS-I**					**SIS-II**								
		RMSE					**RMSE**								
		Key part	**R**	**G**	**B**	**Avg.**	**Key part**	**R**	**G**	**B**	**Avg.**				
Key Sensitivity	N/A	k_1	50.89	85.33	47.81	61.34	k_1	50.77	85.47	47.89	61.38				
		k_2	50.79	85.12	47.74	61.22	k_2	50.83	85.57	47.93	61.44				
		k_3	50.85	85.45	47.89	61.40	k_3	50.84	85.11	47.69	61.21				
		k_4	50.78	85.39	47.81	61.33	k_4	50.84	85.11	47.78	61.24				
		k_5	50.95	85.63	48.01	61.53	k_5	50.93	85.36	47.80	61.36				
		k_6	50.84	85.32	47.86	61.34	k_6	50.90	85.40	47.86	61.39				
															
	PSNR					**PSNR**					**PSNR**				
	Img.	**R**	**G**	**B**	**Avg.**	**Img.**	**R**	**G**	**B**	**Avg.**	**Img.**	**R**	**G**	**B**	**Avg.**
Noise Attacks (25% S& P)	*I*, *R*	33.84	33.82	33.82	33.83	*I*, *R*	33.82	33.82	33.82	33.82	*I*, *R*	33.82	33.82	33.82	33.82
															
	PSNR					**PSNR**					**PSNR**				
	Img.	**R**	**G**	**B**	**Avg.**	**Img.**	**R**	**G**	**B**	**Avg.**	**Img.**	**R**	**G**	**B**	**Avg.**
Crop Attacks (25%)	*I*, *R*	33.91	33.93	33.92	33.92	*I*, *R*	33.92	33.91	33.91	33.91	*I*, *R*	33.93	33.93	33.91	33.93
	Images	**NPCR**	**UACI**	**Images**	**NPCR**	**UACI**	**Images**	**NPCR**	**UACI**						
Differential Attacks	S_1, MS_1	00.000	00.000	S_1, MS_1	00.000	00.000	S_1, MS_1	99.610	33.465						
	S_2, MS_2	00.000	00.000	S_2, MS_2	00.000	00.000	S_2, MS_2	99.605	33.487						
	S_3, MS_3	00.000	00.000	S_3, MS_3	00.000	00.000	S_3, MS_3	99.602	33.450						

The system starts by performing an XOR between the secret image, I, and a random image, L, generated from the Lorenz system. Eight bits are extracted from the x, y, and z states for each pixel to encrypt the R, G, and B channels, respectively. The image E is generated from this substitution stage.

The permutation stage uses the generalized Arnold transform, where the parameters γ and ζ are extracted from the Lorenz system as $\log_2 N$ bits from x and y, respectively. Hence, the image E is permuted to generate the image H. Finally, the image H is shared as n shares using the previously described VSS system.

The recovery scheme for SIS-I is shown in Fig. 2d. First, the n shares are XORed to recover the image H as in the VSS recovery system. Then, the Lorenz system is used to generate the random image L, and extract γ and ζ. The inverse generalized Arnold transform is used to recover the unscrambled image, E. Finally, inverse substitution is applied by performing an XOR between L and E, to recover the secret, R.

Because SIS-I has substitution and permutation stages in addition to the VSS system, it passes all statistical tests. The robustness results are shown in Table 2, where the system passes all the tests except differential attacks, because there is no dependency on the input image. Modification of the system will be introduced in the following section to pass this test.

6 The second SIS system

A dependency on the input image pixels must be present in the second SIS system, SIS-II, to resist differential attacks. This dependency is created by passing the secret image as input to SHA-256 to produce 256 bits digest [33]. The digest is then XORred with the system key to produce a modified key for the Lorenz system. The block diagram of SIS-II generation scehme is shown in Fig. 2e, depicting the addition of SHA-256 to SIS-I.

The digest must be sent to the receiver to generate the same chaotic sequence from Lorenz system. RSA, which is a well-known secured public key cryptosystem, is used to transmit this small digest of 256 bits [34]. It should be noted that RSA is not suitable for encrypting large data, such as images, due to its complexity.

The recovery scheme of SIS-II, as shown in Fig. 2f, is the same as that of SIS-I but with the utilization of the digest as obtained from RSA. The robustness results are shown in Table 2, where the system passes all the tests, giving a secure and robust SIS system.

7 Hardware implementation

The hardware architectures for the three proposed systems are designed and implemented on an FPGA from Xilinix: Genesys2 XC7K325TFFG900-2 [35]. To validate the results and performance, a high-definition multimedia interface (HDMI) display was utilized. In all implementations, 16 bits address busses were used to interface with 65536 depth memories that are suitable for image sizes of 256×256. It is worth mentioning that, to the best of the authors' knowledge, there is no hardware validation in the literature for such SIS systems.

7.1 Main building blocks

7.1.1 Lorenz chaotic system

The architecture of Lorenz chaotic system is shown in Fig. 3a, where it takes the outputs from the Subkeys Generation to generate x, y and z. This block takes x_0, y_0 and z_0 as inputs in the first clock cycle, then the outputs x, y and z are used as the new inputs every new clock cycle. This scheme uses 55-bit fixed point arithmetic, with 1 sign bit, 12 bits for the integer part and 42 bits for the fractional part, which are enough to represent the chaotic system output. The output from each multiplication operation has the size of 110 bits. Only bits 96 down to 42 are taken to keep the integrity of the used 55 fixed point operations. Figure 4a shows oscilloscope projection of 12 bits integer part of Lorenz x-z and Fig. 4b shows x-time projection.

7.1.2 Shares generation

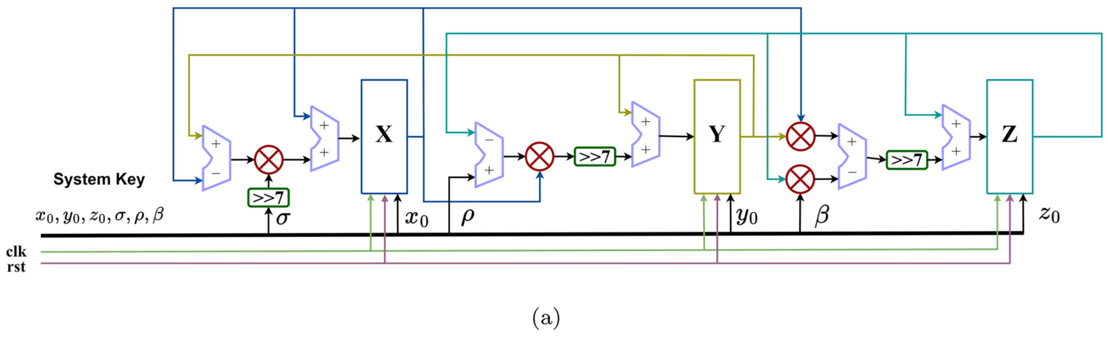
In Fig. 3b, the 8 LSBs of the fractional parts of x, y and z are transmitted between three registers each clock cycle to be saved. The 8 bits x, y and z of each cycle are concatenated to generate the random pixels P_1, P_2 and P_3, where: $P_1 = [x_{t-2}, y_{t-2}, z_{t-2}]$, $P_2 = [x_{t-1}, y_{t-1}, z_{t-1}]$ and $P_3 = [x_t, y_t, z_t]$. After that, the input image pixel P_S is XORed with P_1, P_2 and P_3 to generate P_{Rand}. The 2 LSBs of x_{t-2} are extracted to select the output of each of the four multiplexers, which deliver to the four shares.

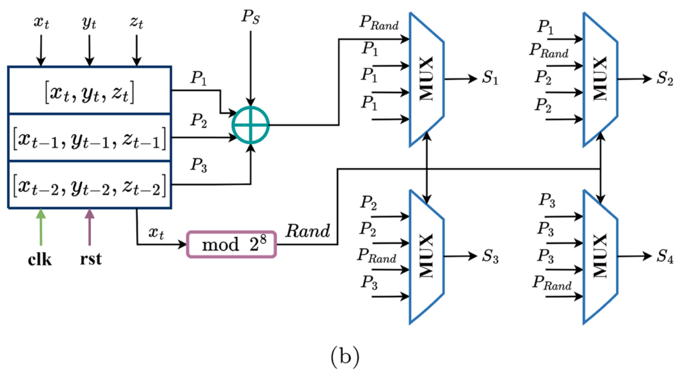
7.1.3 Generalized Arnold transform

In generalized Arnold transform, demonstrated in Fig. 3c, the 16 bits pixel memory address is transformed to a new memory address. Arnold parameters, γ and ζ, are taken from the previously mentioned Lorenz block at a specific clock cycle giving the system extra security. The memory address is initially transformed to the (x, y) coordinates on the screen. Then, γ and ζ are used with simple XOR and addition operations to output (x', y') and, hence, a new memory address. Examples of generalized Arnold transform with different γ and ζ values are given in Fig. 5.

7.1.4 SHA-256 pre-processing

In Fig. 6, SHA-256 is utilized to generate a 256 bits hash. A memory, distinct in both depth, width and address buss from other memories in the system, houses the secret image and yields a 512-bit stream. This stream undergoes





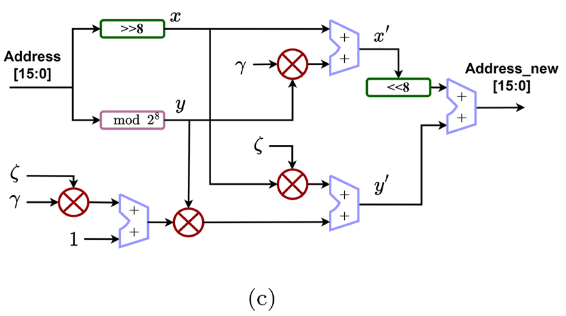


Fig. 3 Hardware architecture of: (**a**) Lorenz system, (**b**) shares generation, and (**c**) generalized Arnold transform

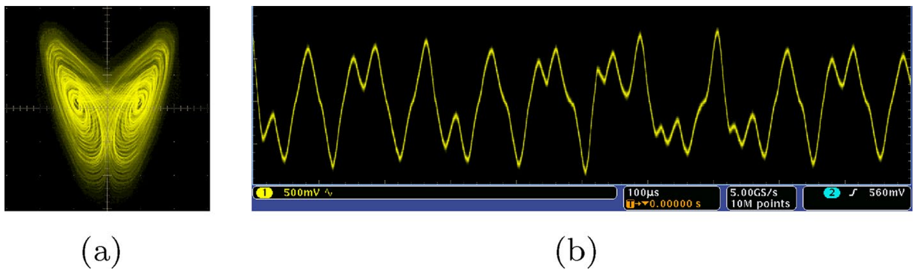


Fig. 4 Lorenz on oscilloscope: (**a**) X–Z projection, (**b**) X-time

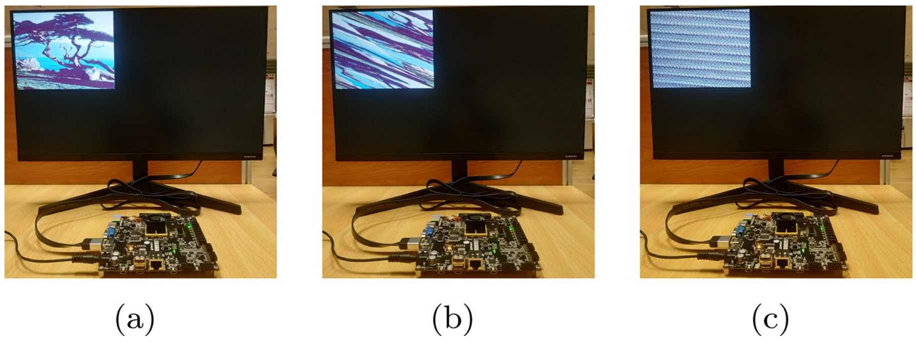


Fig. 5 Generalized Arnold example: (**a**) original image, (**b**) $\gamma = 1,\ \zeta = 1$, and (**c**) $\gamma = 35,\ \zeta = 9$

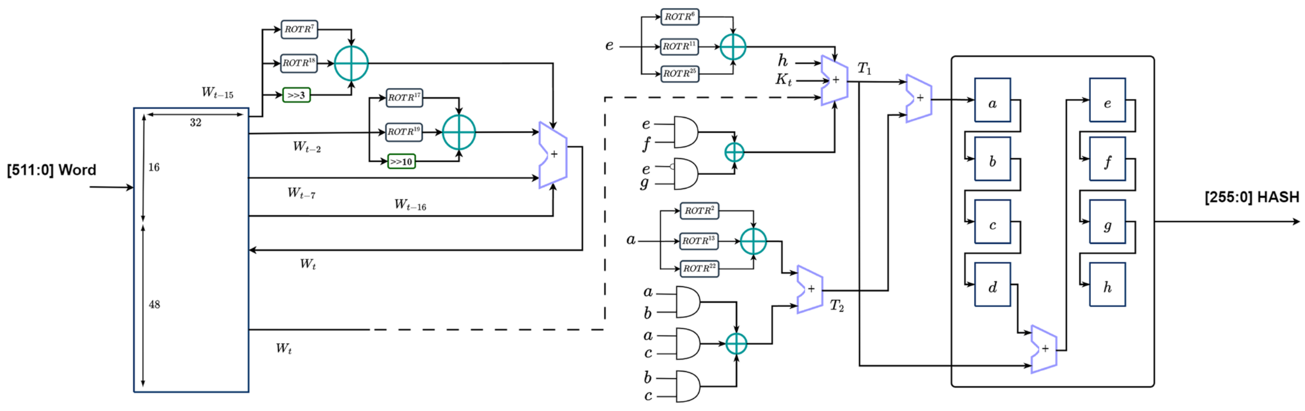


Fig. 6 Hardware architecture of SHA-256

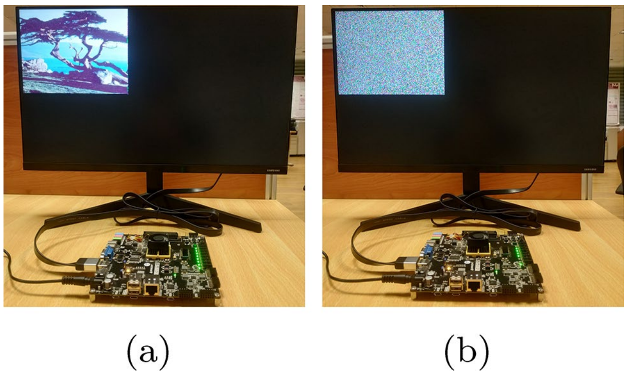


Fig. 7 VSS hardware results of the Genesys2 FPGA implementation showing (**a**) secret image I and (**b**) S_1

restructuring in another smaller memory as shown in the figure. Zero padding is performed to complete the memory vacancy. Then, an iterative process starts to modify the values of these zeros starting from $i = 16$. After this data preparation, starting from $j = 0$, each word is employed to update the values of $[a, b, c, d, e, f, g, h]$ passing through straightforward rotation and XOR operations. Finally, the hash is updated with these values as $H_0 = H_0 + a$ through $H_7 = H_7 + h$.

7.2 Implementation of the VSS system

The generation scheme of the VSS system is mainly based on the Lorenz and shares generation blocks. Lorenz delivers the current x, y and z in each cycle to the shares generation. The process of the shares generation and distribution ends with constructing the four shares of the secret image. The recovery scheme of the VSS system is a straightforward XORing between the four shares to reconstruct the secret image.

Figure 7 shows the secret image and one of the generated shares on a screen as a result from implementing the VSS system. The utilization in Table 3 shows the employment of the FPGA resources. Generation and recovery schemes utilize a large percentage of Block RAMs (BRAMs) as each generated share is stored in addition to the secret and random images. One Mixed-Mode Clock Manager (MMCM) was utilized to generate the system clock. The employed Inputs and Outputs (IO) are mainly for the HDMI module in addition to a number of switches and LEDs for system interface.

Table 3 Hardware utilization

Resource	Available	VSS				SIS-I				SIS-II			
		Generation		Recovery		Generation		Recovery		Generation		Recovery	
		Util	Util.(%)	Util	Util.(%)	Util	Util.(%)	Util	Util.(%)	Util	Util.(%)	Util	Util.(%)
LUT	203800	1766	0.87	742	0.36	2130	1.05	2032	1.00	9792	4.80	2076	1.02
FF	407600	661	0.16	241	0.06	755	0.19	687	0.17	3110	0.76	743	0.18
BRAM	445	220	49.44	220	49.44	308	69.21	352	79.10	358.50	80.56	352	79.10
DSP	840	36	4.29	0	0.00	36	4.29	36	4.29	36	4.29	37	4.40
IO	500	24	4.80	20	4.00	25	5.00	20	4.00	25	5.0 0	20	4.00
MMCM	10	1	10.00	1	10	1	10	1	10.00	1	10	1	10

Lookup tables (LUTs), Flip Flops (FF) and Digital Signal Processors (DSP) percentages are relatively small.

7.3 Implementation of SIS-I

Leveraging the Lorenz, shares generation and Arnold blocks, three consequent stages are needed for SIS-I generation. In the first stage, the Lorenz block initiates a random generation process to form the random image ’L’ mirroring the dimensions of the secret image. A bitwise XOR operation is then employed on ’L’ and the secret image to produce the ’E’ image. In the second stage, Lorenz transmits the current 8 LSBs of ’x’ and ’y’ values to the Arnold block, serving as ’γ’ and ’ζ’ parameters. Arnold subsequently transforms the memory address, generating a new writing address for the image ’H’. In the third stage, the ’H’ image undergoes the same process as in the VSS system going through bitwise XORing with the random pixels to finally generate the four shares.

The recovery scheme of SIS-I starts with XORing the four shares to get the ’H’ image in parallel with operating Lorenz to build the ’L’ image. Afterward, the inverse Arnold transform takes its parameters from Lorenz to reconstruct the image ’E’ from the ’H’ image. The ’L’ image stage must come before the inverse Arnold stage to match the same γ and ζ of the generation scheme. Finally, ’L’ and ’E’ images are XORed to recover the secret image.

The experimental results of SIS-I look similar to those given in Fig. 7. As shown in Table 3, the system utilizes more BRAMs in its generation and recovery schemes than the VSS system due to the addition of the random images ’H’, ’E’ and ’L’. In the generation process, however, ’L’ does not need to be stored in a memory as it is XORed with the original image directly to generate and store the ’E’ image. In the recovery process, this cannot be done as the XOR operation is the final step.

7.4 Implementation of SIS-II

As depicted in Fig. 8a, the generation process of SIS-II uses the four main building blocks. The initial stage involves pre-processing, wherein the system key undergoes modification using the SHA-256 block. The resultant 256-bit digest is then XORed with the system key to yield the modified key. Then, Lorenz starts with the new modified key, and the generation process continues as in SIS-I.

In SIS-II, the recovery scheme shown in Fig. 8b is almost identical to that of SIS-I except for the initial modification of the system key. The original system key is XORed with the SHA-256 digest to get the same modified system key used in the generation scheme.

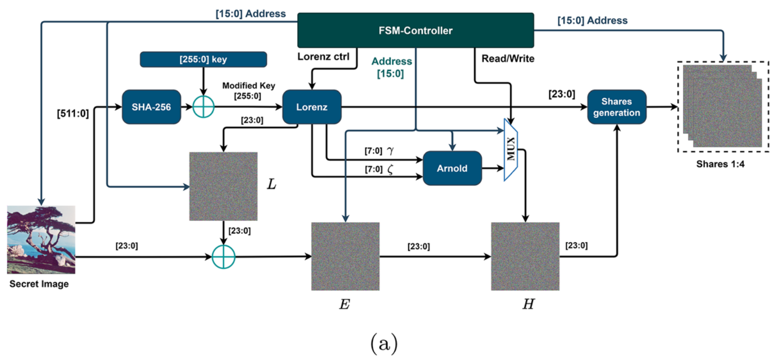
The utilization of SIS-II given in Table 3 shows that BRAM and LUT utilization in the generation scheme are more than the previous system due to the addition of SHA-256 pre-processing. The recovery utilization is exactly as in SIS-I, where the only difference between them is the starting system key.

8 Results and comparisons

The images used in evaluating the three systems are from the USC-SIPI image database as shown in Table 4 with their names, descriptions, and sizes [36]. The Tree image is used to show the software and hardware results of the three systems in detail when $n = 4$. Additional detailed results for all the images of Table 4 are provided in the supplementary document with different n values and similar good results.

The histograms of the secret I and the first share S_1, from SIS-II hardware, are given in Fig. 9, where all the shares in the three systems give similar uniform histograms indicating good encryption.

Table 5 shows the average security analysis results of the software and hardware for Tree when $n = 4$ for the three



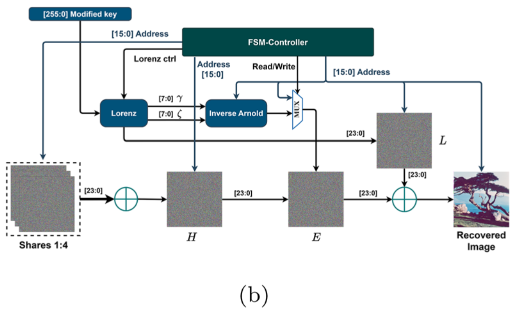


Fig. 8 Architecture diagram of SIS-II: (**a**) generation scheme and (**b**) recovery scheme

Table 4 Used images from USC-SIPI database [36]

Image			
Name	4.1.06	4.1.08	4.2.03
Description	Tree	Jelly beans	Mandrill
Size	256 × 256		512 × 512
Image			
Name	4.2.07	2.2.05	2.2.07
Description	Peppers	San Diego	Oakland
Size	512 × 512	1024 × 1024	

proposed systems, where the detailed results are given in the supplementary document.

Even though the software implementation uses floating-point arithmetic and the hardware implementation uses fixed-point arithmetic, the table demonstrates that both implementations provide good and comparable security measures. The values of the hardware and software results are too close to each other but different due to the different implementations. More detailed results for the different channels are given in the supplementary file.

The entropy results in the produced shares for the three systems are close to 8, indicating good randomness. The RMSE and correlation values between the secret image and the shares in the three systems give high RMSE and low correlation values, indicating good encryption. The correlation results between the adjacent pixels in the shares are low in the horizontal, vertical, and diagonal directions in the three systems. Also, the adjacent pixels scatter diagrams in the vertical direction of the secret and the first share for SIS-II are shown in Fig. 10, where all the shares in the three systems in different directions give similar no correlation diagrams indicating good encryption.

NIST test results of S_1 for SIS-II are given in Table 6 with $\alpha = 0.001$, where all the shares of the three systems give similar passing results. A secret image of size 1024 × 1024 is used to satisfy the NIST requirement of at least one million numbers.

The key sensitivity tests were conducted on the different parts of the system key by changing the LSB of each part separately, and the results for SIS-I and SIS-II were calculated since VSS does not use the key in the recovery stage. The results of the RMSE are given in Table 2, showing good key sensitivity results. These results also show that different shares can be produced for the same image if the key changes.

The BER is also measured between the original image and the recovered image when one bit is changed in different locations of the key [37]. Figure 11 shows BER results near 0.5, which is the desired value, demonstrating good key sensitivity and that no partial information can be revealed about the original image [38].

Noise attack results are the same for the three systems, as shown in Table 2, where salt and pepper noise with intensity 25% was added to the first share. The resulting PSNR values indicate good resistance to noise attacks.

Crop attack results are better for SIS-I and SIS-II because of the presence of the permutation stage, which plays an important role in passing crop attacks. The software crop attack results for the three systems are shown in Table 2,

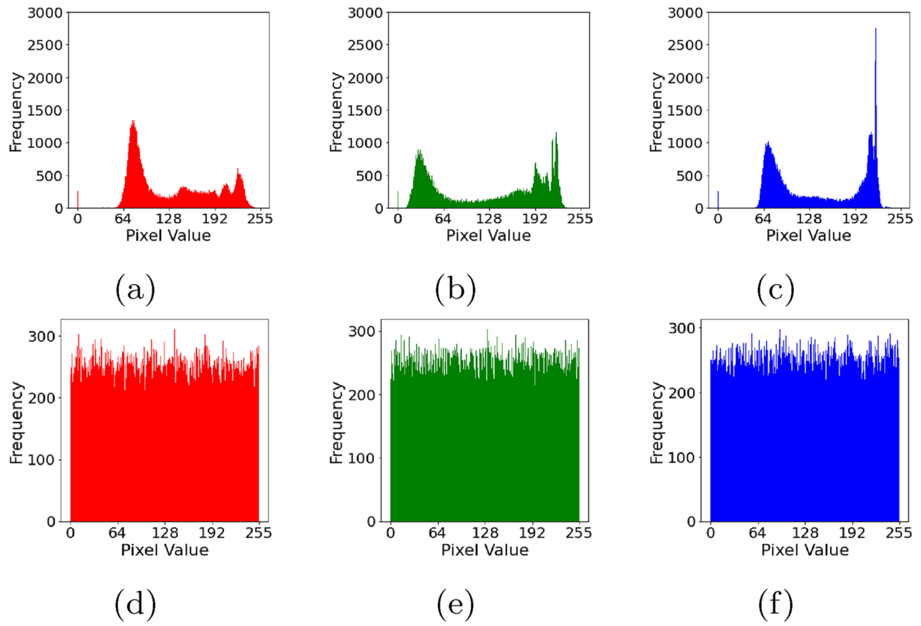
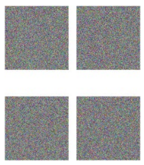
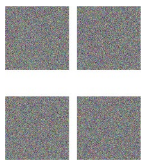
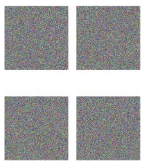


Fig. 9 Histograms of (**a**)–(**c**) Tree, and (**d**)–(**f**) S_1 from SIS-II hardware when $n = 4$

Table 5 Security analysis results for software and hardware of Tree when $n = 4$

Test	VSS		SIS-I		SIS-II	
Shares						
Images	SW	HW	SW	HW	SW	HW
Entropy						
S_1	7.9971	7.9974	7.9975	7.9974	7.9973	7.9971
S_2	7.9973	7.9973	7.9973	7.9971	7.9971	7.9971
S_3	7.9970	7.9972	7.9972	7.9971	7.9969	7.9971
S_4	7.9973	7.9970	7.9972	7.9972	7.9973	7.9969
Adjacent pixels correlation						
S_1	0.0005	0.0007	0.0026	−0.0013	0.0001	0.0016
S_2	0.0009	0.0003	−0.0012	−0.0008	0.0012	−0.0005
S_3	0.0006	0.0001	0.0016	−0.0001	−0.0028	0.0013
S_4	0.0004	0.0004	0.0011	0.0013	−0.0014	0.0008
Correlation						
I, S_1	0.0010	0.0012	0.0018	0.0002	−0.0000	0.0038
I, S_2	0.0013	0.0011	0.0019	−0.0042	0.0005	0.0023
I, S_3	−0.0016	−0.0001	−0.0007	0.0009	−0.0045	0.0050
I, S_4	0.0007	0.0023	−0.0027	−0.0011	0.0001	0.0003
S_1, S_2	−0.0032	0.0022	−0.0043	0.0023	0.0026	0.0009
S_1, S_3	0.0019	0.0040	−0.0038	0.0035	0.0047	−0.0031
S_1, S_4	−0.0041	0.0014	0.0018	−0.0012	−0.0004	0.0001
S_2, S_3	−0.0007	0.0007	0.0059	−0.0008	0.0009	−0.0020
S_2, S_4	−0.0002	−0.0007	−0.0002	0.0019	0.0017	−0.0000
S_3, S_4	0.0029	0.0009	0.0019	0.0005	−0.0008	0.0021
RMSE						
I, S_1	99.42	99.43	99.43	99.48	99.51	99.27
I, S_2	99.40	99.42	99.35	99.61	99.33	99.46
I, S_3	99.61	99.58	99.50	99.44	99.73	99.21
I, S_4	99.48	99.41	99.59	99.61	99.53	99.50
S_1, S_2	104.63	104.35	104.73	104.28	104.24	104.48
S_1, S_3	104.42	104.37	104.69	104.29	104.31	104.59
S_1, S_4	104.70	104.42	104.41	104.63	104.58	104.51
S_2, S_3	104.57	104.54	104.13	104.44	104.32	104.61
S_2, S_4	104.51	104.53	104.46	104.40	104.28	104.59
S_3, S_4	104.41	104.56	104.34	104.54	104.60	104.39

while the recovered images from the hardware are given in Fig. 12. Although the PSNR values are the same in the three systems, because the amount of data lost is the same, the VSS system does not pass the test visually.

Only SIS-II resists differential attacks due to the presence of SHA-256, which builds the dependency on the input image. Table 2 shows the results, where SIS-II gives the optimal values for UACI and NPCR.

Runtime is measured for the three systems with different numbers of shares and images of size 512×512. The average of fifty runs is given in Table 7, where the number of shares slightly affects the recovery time. In generation, the modifications on the VSS only add one second of runtime for SIS-I and SIS-II systems, while maintaining the fast recovery. The used setup is (Windows 11 Pro, Intel(R) Core(TM) i7-8750 H CPU @ 2.20 GHz, 15.8 GB RAM) using Python programming language on JupyterLab IDE.

To compare the proposed SIS-II with previous techniques, it was adopted for grayscale images and compared with the system in [24], for the used grayscale Mandrill

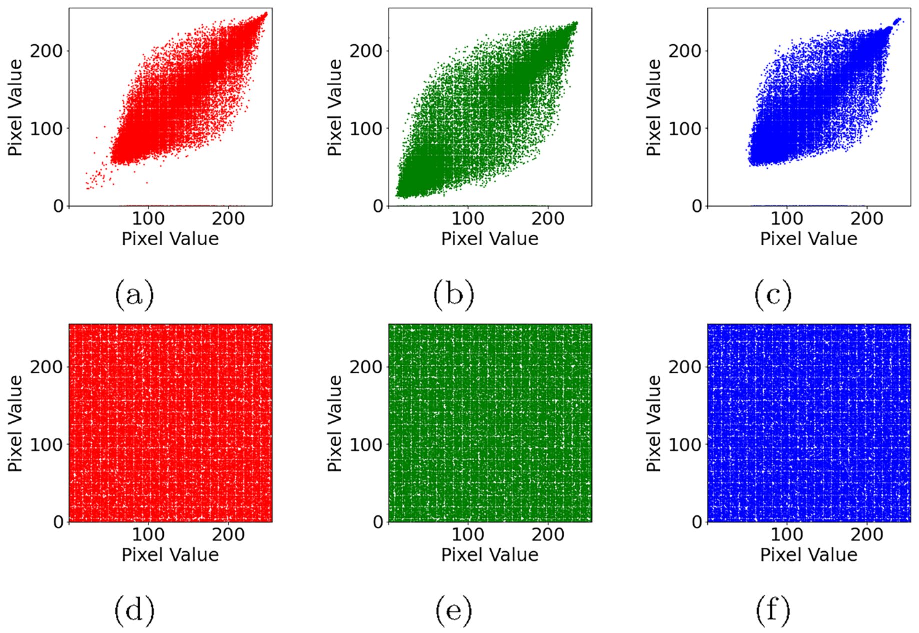


Fig. 10 Scatter diagrams between adjacent pixels in the vertical direction of (**a**)–(**c**) Tree, and (**d**)–(**f**) S_1 for SIS-II hardware when $n = 4$

Table 6 NIST results of S_1 for SIS-II

Test	P-Value	PP	Result
Frequency	0.162606	0.958	✓
Block Frequency	0.002043	0.958	✓
Cumulative Sums	0.523809	0.958	✓
Runs	0.350485	1.000	✓
Longest Run	0.025193	1.000	✓
Rank	0.437274	1.000	✓
FFT	0.350485	1.000	✓
Non Overlapping Template	0.332769	0.988	✓
Overlapping Template	0.066882	0.958	✓
Universal	0.048716	1.000	✓
Approximate Entropy	0.090936	1.000	✓
Random Excursions	0.124944	0.992	✓
Random Excursions Variant	0.099629	0.996	✓
Serial	0.342918	1.000	✓
Linear Complexity	0.122325	1.000	✓

image. The comparison is given in Table 8, where the results are good.

9 Conclusions

Three systems were proposed for sharing any type of image, starting with the VSS system as the basic module to ensure a fast recovery process. Then, SIS-I used the Lorenz chaotic system as the source of randomness, and the generalized Arnold transform as a permutation module. Finally, SIS-II added extra levels of security by utilizing SHA-256 and RSA. Moreover, FPGA architectures were designed and implemented to boost computational efficiency and enable seamless real-time processing. The experimental results validated the effectiveness and practicality of these implementations with minimal resource utilization. Security analysis and comparisons with related literature were presented with good results including statistical tests, differential attack measures, robustness tests against noise and crop attacks, key sensitivity tests, and performance analysis. Other permutation algorithms, chaotic systems, and VSS systems can be further investigated to find the best combination for creating SIS systems.

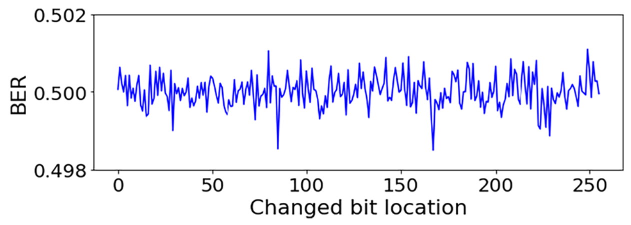


Fig. 11 BER between the original and recovered images using different keys with a modified bit in different locations

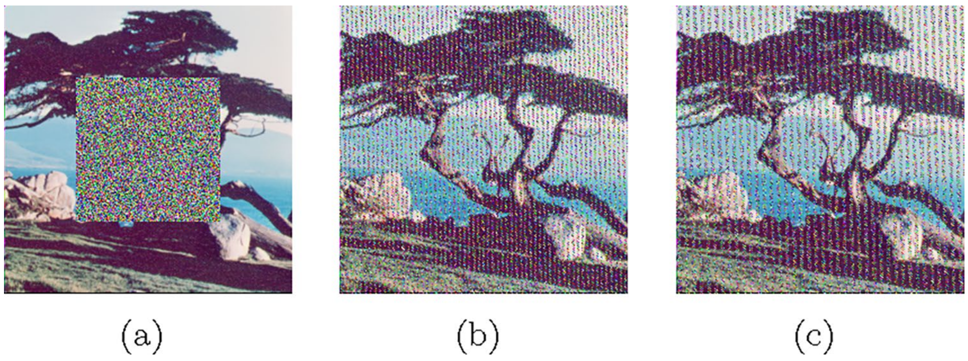


Fig. 12 Recovered images from hardware after 25% crop attack: (**a**) VSS, (**b**) SIS-I, and (**c**) SIS-II

Table 7 Runtime, in seconds, for color images of size 512 × 512 and different numbers of shares, n

n	VSS		SIS-I		SIS-II	
	Gen.	Rec.	Gen.	Rec.	Gen.	Rec.
2	1.285	0.001	2.249	0.484	2.257	0.486
3	1.985	0.002	2.907	0.484	2.916	0.486
4	2.633	0.002	3.548	0.484	3.549	0.486
5	3.301	0.003	4.174	0.484	4.188	0.486

Table 8 Comparison with previous scheme using grayscale Mandrill with n = 4

		RMSE		Correlation		PSNR		UACI		NPCR	
Images		[24]	Prop	[24]	Prop	[24]	Prop	[24]	Prop	[24]	Prop
I_i, S_i	I_1, S_1	84.7894	**85.2110**	0.0045	**0.0002**	9.5640	**9.5209**	–	–	–	–
	I_2, S_2	85.2877	**85.3137**	− 0.0048	**− 0.0023**	9.5131	**9.5104**	–	–	–	–
	I_3, S_3	**85.4143**	85.0411	−0.0084	**0.0034**	**9.5002**	9.5382	–	–	–	–
	I_4, S_4	85.1378	**85.2989**	**0.0001**	−0.0030	9.5284	**9.5119**	–	–	–	–
S_i, S_j	S_1, S_2	102.8805	**108.4050**	**0.0258**	−0.0293	7.8841	**7.4298**	32.97	34.94	99.99	99.02
	S_1, S_3	**109.6515**	102.4483	**0.0106**	0.0959	**7.3305**	7.9207	35.12	32.45	99.88	99.80
	S_1, S_4	101.2383	**108.8057**	0.0571	**− 0.0300**	8.0239	**7.3978**	31.95	35.19	99.89	99.41
	S_2, S_3	105.4132	**105.5462**	− 0.0202	**− 0.0068**	7.6729	**7.6620**	33.75	33.65	99.99	99.02
	S_2, S_4	**106.3458**	102.0771	**− 0.0376**	0.0488	**7.5964**	7.9522	33.90	31.91	98.22	98.63
	S_3, S_4	**106.2978**	105.8779	− 0.0369	**− 0.0092**	**7.6003**	7.6347	34.35	33.77	99.98	99.22

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11554-024-01450-8.

Acknowledgements We thank Marwan A. Fetteha (Nile University) and Abdallah S. Salem (Nile University) for their support in the hardware implementation of the Lorenz chaotic system and VSS system.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability Images used in testing the proposed systems are publicly available: https://sipi.usc.edu/database/ Ref. [36].

Declarations

Funding This work is supported by the Science, Technology, and Innovation Funding Authority (STIFA), Egypt, under grant number 45631.

Conflict of interest The authors declare that they have no Conflict of interest.

References

1. Ibrahim, D.R., Teh, J.S., Abdullah, R.: An overview of visual cryptography techniques. Multimedia Tools Appl. **80**, 31927-31952 (2021). (**9**)
2. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
3. Naor, M., Shamir, A.: Visual cryptography. Advances in Cryptology – EUROCRYPT'94 (1995)
4. Li, P., Yang, C.-N., Kong, Q.: A novel two-in-one image secret sharing scheme based on perfect black visual cryptography. J. Real-Time Image Process. **14**(1), 41–50 (2016)
5. Thien, C.-C., Lin, J.-C.: Secret image sharing. Comput. Graph. **26**(5), 765–770 (2002)
6. Elsafty, A.H., Tolba, M.F., Said, L.A., Madian, A.H., Radwan, A.G.: Enhanced hardware implementation of a mixed-order nonlinear chaotic system and speech encryption application. AEU-Int. J. Electron. Commun. **125**, 153347 (2020)
7. Ravichandran, D., Rajagopalan, S., Upadhyay, H.N., Rayappan, J.B.B., Amirtharajan, R.: Encrypted biography of biomedical image-a pentalayer cryptosystem on fpga. J, Signal Process. Syst. **91**, 475–501 (2019)
8. Monmasson, E., Cirstea, M.N.: Fpga design methodology for industrial control systems-a review. IEEE Trans. Ind. Electron. **54**(4), 1824–1842 (2007)
9. Shivani, S., Patel, S.C., Arora, V., Sharma, B., Jolfaei, A., Srivastava, G.: Real-time cheating immune secret sharing for remote sensing images. J. Real-Time Image Process. **18**(5), 1493–1508 (2020)
10. Wan, S., Qi, L., Yang, G., Lu, Y., Yan, X., Li, L.: Visual secret sharing scheme with (n, n) threshold for selective secret content based on QR codes. Multimedia Tools Appl. **79**(3–4), 2789–2811 (2019). (**12**)
11. Liu, Z., Zhu, G., Ding, F., Kwong, S.: Weighted visual secret sharing for general access structures based on random grids. Signal Process. **92**, 116129 (2021). (**3**)
12. Chiu, P.-L., Lee, K.-H.: Threshold visual cryptography schemes with tagged shares. IEEE Access **8**, 111 330-111 346 (2020)
13. JBA, Raj, C, Sukumaran, R., S.M.G.: Enhanced semantic visual secret sharing scheme for the secure image communication. Multimedia Tools and Applications, **79**(23-24), 17 057–17 079, 4 (2019)
14. Wang, L., Yan, B., Yang, H.-M., Pan, J.-S.: Flip extended visual cryptography for gray-scale and color cover images. Symmetry **13**, 65, 12 (2020)
15. Kannojia, S.P., Kumar, J.: XOR-based visual secret sharing scheme using pixel vectorization. Multimedia Tools Appl. **80**(4), 14609-14635 (2021)
16. Sridhar, S., Sudha, G.F.: Quality improved (k, n) priority based progressive visual secret sharing. Multimedia Tools Appl. **79**(17–18), 11459-11486 (2020). (**1**)
17. Li, P., Ma, J., Ma, Q.: (t, k, n) XOR-based visual cryptography scheme with essential shadows. J. Visual Commun. Image Representation **72**, 102911, 10 (2020)
18. Patil, S.M., Purushothama, B.: Pixel co-ordinate-based secret image sharing scheme with constant size shadow images. Comput. Electr. Eng. **89**, 106937 (2021)
19. Xiong, L., Zhong, X., Yang, C.-N., Han, X.: Transform domain-based invertible and lossless secret image sharing with authentication. IEEE Trans. Inform. Forensics Secur. **16**, 2912–2925 (2021)
20. Sharobim, B.K., Fetteha, M.A., Abd-El-Hafiz, S.K., Sayed, W.S., Said, L.A., Radwan, A.G.: An efficient multi-secret image sharing system based on Chinese remainder theorem and its FPGA realization. IEEE Access **11**, 9511–9520 (2023)
21. Chen, J., Liu, K., Yan, X., Liu, L., Zhou, X., Tan, L.: Chinese remainder theorem-based secret image sharing with small-sized shadow images. Symmetry, **10**(8), (2018)
22. Liu, Y.N., Zhong, Q., Xie, M., Chen, Z.B.: A novel multiple-level secret image sharing scheme. Multimedia Tools Appl. **77**, 6017–6031 (2018). (**3**)
23. Cheng, Y., Fu, Z., Yu, B.: Improved visual secret sharing scheme for QR code applications. IEEE Trans. Inform. Forensics Secur. **13**, 2393–2403 (2018). (**9**)
24. Pande, D., Rawat, A.S., Deshmukh, M., Singh, M.: Single secret sharing scheme using chinese remainder theorem, modified shamir's scheme and XOR operation. Wirel. Personal Commun. **130**(2), 957–985 (2023). (**3**)
25. Lorenz, E.N.: Deterministic Nonperiodic Flow. J. Atmospheric Sci. **20**, 130–141 (1963). (**3**)
26. Butcher, J.C.: Numerical Methods for Ordinary Differential Equations. Wiley (2016). (**7**)
27. Mansingka, A., Affan Zidan, M., Barakat, M., Radwan, A., Salama, K.: Fully digital jerk-based chaotic oscillators for high throughput pseudo-random number generators up to 8.77gbits/s. Microelectron. J. **44**(9), 744–752 (2013). https://doi.org/10.1016/j.mejo.2013.06.007
28. Liu, H., Zhao, B., Huang, L.: Quantum image encryption scheme using arnold transform and s-box scrambling. Entropy **21**(4), 343, 3 (2019)
29. Dang, Q.H.: Secure hash standard. National Institute of Standards and Technology, Tech. Rep., (Jul. 2015)
30. Bassham, L., Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Leigh, S., Levenson, M., Vangel, M., Heckert, N., Banks, D.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. 2010-09-16 (2010)
31. Sharobim, B.K., Abd-El-Hafiz, S.K., Sayed, W.S., Said, L.A., Radwan, A.G.: Pixel-based visual secret sharing using lorenz system. In: Proceedings of the 2023 8th International Conference on Cloud Computing and Internet of Things, ser. CCIOT '23. New York, NY, USA: Association for Computing Machinery (2023)
32. Abd-El-Hafiz, S.K., AbdElHaleem, S.H., Radwan, A.G.: Novel permutation measures for image encryption algorithms. Opt. Lasers Eng. **85**, 72–83 (2016). (**10**)
33. Wang, J., Liu, G., Chen, Y., Wang, S.: Construction and analysis of SHA-256 compression function based on chaos s-box. IEEE Access **9**, 61 768-61 777 (2021)
34. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978). (**2**)
35. Digilent: "Genesys 2 FPGA Board Reference Manual," Digilent, (2017). [Online]. https://digilent.com
36. Weber, A.: The USC-SIPI image database. Signal Image Process. Inst. Univ. South. California., (1997). [Online]. https://sipi.usc.edu/database/
37. Merah, L., Adnane, A., Ali-Pacha, A., Ramdani, S., Hadj-said, N.: Real-time implementation of a chaos based cryptosystem on low-cost hardware. Iran. J. Sci. Technol. Trans. Electr. Eng. **45**(4), 1127–1150 (2021). https://doi.org/10.1007/s40998-021-00433-w
38. Merah, L., Ali-Pacha, A., Hadj-Said, N.: Real-time cryptosystem based on synchronized chaotic systems. Nonlinear Dyn. **82**(1–2), 877–890 (2015). https://doi.org/10.1007/s11071-015-2202-2