IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 16, 2021, accepted January 26, 2022, date of publication February 7, 2022, date of current version February 11, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149424

Secure Real-Time Chaotic Partial Encryption of Entropy-Coded Multimedia Information for Mobile Devices: Smartphones

ROGELIO HASIMOTO-BELTRAN, (Member, IEEE), MARCOS D. CALDERON-CALDERON, AND VÍCTOR H. OLAVARRÍA-JARAMILLO

Center for Research in Mathematics (CIMAT), A.C., Guanajuato, Gto 36023, México

Corresponding author: Rogelio Hasimoto-Beltran (hasimoto@cimat.mx)

This work was supported by the Center for Research in Mathematics (CIMAT).

ABSTRACT Smartphones penetration-rate continues expanding, from 44% in 2017 up to 59% expected increase in 2022 as reported by Strategy Analytics. At present, smartphones dominate the global mobile traffic, which in turn is dominated by video communications. For mobile systems with limited power capabilities, the processing of real-time multimedia information with affixed security represents a real challenge. In this work, we propose a high-performance encryption scheme capable of running on low-power smartphones without holding back video coding operation. We are aimed at destroying the meaning of the entropy coded bitstream by inserting random bit errors to induce error propagation and impede natural self-resynchronization process. The scheme consists of three main process: 1) A new integer *chaos-based Coupled-Map Lattice*-CML for creating secure pseudo-random trajectories; 2) *Random bit flipping* of the bitstream based on a Dynamic Reference Point (DRP) not exposed to attackers; and 3) *Random selection* of CML byte-trajectories for both *DRP* and bit flipping processes for increased security. Implementation of the scheme on smartphones with different CPU-power capabilities shows *excellent performance* to handle high-bandwidth real-time video smoothly (one-to-one and group calls). The scheme provides high scalable security with encrypted data volume fluctuating between 0.7%-3% of the total compressed data (video and still images).

INDEX TERMS Chaotic systems, coupled-map lattice, entropy coding, partial encryption.

I. INTRODUCTION

The number of mobile phones users worldwide has increased almost 600% during the last decade, going from 1.06 billion users in 2012 to 6.64 billion in 2022 [1]. Similarly, the global internet traffic share (dominated by desktop users until 2016) rose in such a way that, as of July 2021, 55.56% of all web traffic comes through mobile phones [2]. One of the main activities of mobile users is on-line video, accounting for 64% in 2014 and projected to grow up to 82% in 2021, representing a million minutes of video content crossing the network every second [3]. Digital communications have some inherent risks, communication networks (wired or wireless) are vulnerable to attacks violating the user's right of privacy.

Compared to other communication media datatype such as image, audio, graphics and text, securing real-time video data demands the addressing of hard challenges on mobile devices, among the most important are: a) *the vast amount of information involved*; b) *coding complexity*; c) *time processing constraints*; and d) *limited processing power*, in particular low-to-mid-range smartphones, which dominate the market worldwide according to IDC (International Data Corporation) [51]. To avoid the processing overhead of full-data encryption, Partial or Selected Encryption (SE) has been proposed as a viable alternative [4]–[9] in which, only the most important information is encrypted. Relevant data involves our human perception of a particular media datatype (i.e. audio, image, video, etc.), which is transferred during compression to transform coefficients (DCT-based, Wavelet-based, etc.) and subsequent bitstream via data transformations and variable-length entropy coding (Huffman and Arithmetic Codes) respectively. The main advantages of SE with respect to full-encryption are: a) *scalability*; b) *fast performance* (it varies depending on the complexity

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

of the scheme); *reasonable-to-high security* (depending on the method and percentage of the encrypted bitstream); and c) *tunable perception quality* for partial content exposure.

SE can be applied to different data types: a) *Plain data* (uncompressed domain without further compression), b) *Transform data* during the compression process, called *Joint Compression-Selective Encryption-JCSE*, and c) *Encoded data* (after compression) [10]. SE over plain data has been particularly applied to plain images (without compression), wherein relevant information (image bit-planes [11], [12] or regions of interest [13], [14]) that contributes the most to the perceptual quality of the object in question is subject for encryption. Even though these schemes consider a subset of plain data, there is no performance gain with respect to full encryption in the compress domain (they may end up encrypting higher percentage of data). Encrypting plain data is not recommended in practice because the compression ratio is severely affected. *JCSE* on the other hand, has been successfully applied to all classes of multimedia objects such as audio, image, and video. The main target are transform-coefficients and encoded bitstream. In speech applications using the G.729 codec for example, the most valuable part of the bitstream subject for encryption includes vector quantization indices, line spectral frequencies, quantized pitch period, and gain indices [7], [15]. MPEG4-Audio on the other side offers scalable embedded coding, in which the lower rate base layer represents the intelligible part of the speech to be encrypted [16]. The amount of encrypted data in speech coding is significantly reduced to about 3%-45% of the total bitstream, however, not all JCSE schemes fall in the category of strong encryption [17]. The vast majority of JCSE schemes in the literature are aimed at securing image and video data. Relevant data to be encrypted may include one or more of the following data blocks: transform coefficients, headers, Intra-frames, motion compensation, entropy coding tables and indices, base and enhancement layers in scalable coding, etc.

SE schemes over DCT-based compression standards (JPEG, MPEG-4, H.26X, etc.) are vast. They can be applied to practically any intermediate element during the encoding process or even after encoding, as in our actual proposal. In [18], a pseudorandom DCT-sign change and entropy coding bit inversion to protect Regions of Interest (ROI) is proposed; authors in [19] and [20] considered DC and AC shuffling only; a more complex scheme is presented in [52], where a subset of DC and AC coefficients are scrambled after zig-zag scanning and encrypted. Intra and Inter macroblocks encryption is considered in [21]; an optimized SE is applied to motion vectors and quantized coefficients in [53]; in [54], [57], authors applied SE at the CABAC bitstream (MVD signs, NON-ZERO TC, etc.) in the scalable extension of the High Efficiency Video Coding (HEVC) standard; a fast video encryption for smartphones is proposed in [55], which encrypts only the DC/ACs of I-macroblocks and the motion vectors of P-macroblocks (taking advantage of the error propagation of the H.264); video slices are encrypted in [56] with two different encryption schemes, AES cipher-feedback mode to maintain the exact same bit rate and a four-dimensional hyperchaotic algorithm for further protection; residual data such as runs and level are encrypted in [22]. In the case of DCT-based scalable compression, different levels of information can be distinguished and selectively protected, such as base and enhancement layers, temporal scalability and/or spatial/SNR scalability [23]. On wavelet-based formats (JPEG2000, SPHIT-Set Partitioning in Hierarchical Trees, etc.), selective encryption is commonly applied to low resolution sub-bands, wherein the most energy is concentrated on [10], [24]–[27], [58]. Significant parameters such as sign bits, refinement bits, significance of pixels, etc., can also be relevant for data encryption, providing different degrees of security [27]. Within the same JCSE category, some others encryption schemes secure the entropy coded bitstream (inside the codec) in a variety of ways: i) codewords are replaced by other valid and equal length codewords in order to maintain video compliance without affecting the compression ratio [9]; ii) selected intervals in the bitstream are randomly swapped (using *randomized arithmetic coding* (RAC), so that only a synchronized decoder is able to interpret correctly the encoded sequence [5]; iii) bitstream is encoded using multiple and randomly selected Huffman tables [4], and iv) partitioned of the bitstream into random blocks followed a circular random rotation within each block [6].

Major concerns of JCSE schemes are: a) codec intrusive; b) codec specific, implying strong dependencies on both multimedia format (such as H.264, MPEG4, JPEG, JPEG2000, etc.) and multimedia object (image, audio, video, etc.); and c) negative effect on the compression ratio (except when the encryption is performed in the entropy coded bitstream without changing its length). As a response to these concerns, we propose a *codec independent* partial encryption scheme with the following properties: *non-intrusive (performed after the coding process)*, *non-selective*, and consequently *format independent*. We take advantage of the Variable-Length Coding (VLC) sensitivity to bit errors in order to induce a loss in synchronization and impede at the same time the long-term self-synchronization capability of entropy coded bitstreams [28]. The frequency of induced bit-errors (or random bit flipping) depends on the Mean Error Propagation length (MEPL) [29], which represents the number of affected codewords (error propagation) until self-synchronization occurs. Our new scheme, derived from our previous proposed scheme in [17], proposes major changes that considerably *simplify* the encryption process, *improve* performance, *increase* security, *and* more importantly provide *computational stability* (see Wobbling effect below). We are aimed at providing high performance and secure codec-independent SE for low-to-mid power smartphones based on three major contributions: a) New integer-based *Pseudo-Random Number Generator* (PRNG) for secure random bit errors in the bitstream; b) a clueless *phantom Dynamic Bit-Reference Point (DRP)* for the bit flipping; and c) *Random selection* of byte-trajectories for the both *DRP* and bit flipping processes (this step

doubles the security of our previous scheme [17]). The justification of our new proposal is as follows. Our new PRNG (based on Coupled Map Lattice-CML) works entirely in the integer domain. With this new integer representation, we elude both floating-point arithmetic (eliminating high computational costs) and the so-called Wobbling Floating-Point Precision (WFP). WFP demands perfect coherence between cipher and decipher, otherwise the inversion of encryption cannot always be guaranteed [30]. Despite the use of IEEE 754 floating-point compliance, several aspects of floating-point operations depend on the system designer which may yield different results for the same computation on two different computers. By using integer-based CML, our partial encryption scheme becomes more accessible to mobile handheld devices, improving the performance of high-level languages such as Java (which is the main development platform on Android OS). Secondly, and considering that, the high-complexity of our previous scheme was to repel known/chosen plaintext attacks to the bit flipping and shuffling process, we propose a new version of the bit flipping that makes use of a phantom *DRP,* that works as a variable reference location for the flipping of bits in the bitstream. *DRP* is not directly exposed under a known/chosen attack, hence the name of phantom *DRP*. Furthermore, we eliminate the shuffling process proposed in [17] (not robust to chosen/plaintext attack) and include an optional and secure step in which chaotic trajectories are randomly selected in such a way that, they may come from N different maps and from different iterations in time. Our scheme duplicates the security of [17], showing extremely fast speed on different smartphone CPU technologies and Android OS versions. To the best of our knowledge, our proposed scheme provides the best performance speeds reported in the literature for smartphones.

The rest of the paper is organized as follows. The next section describes the methodology, including our previous and current schemes for comparison purpose. Security and performance of our proposed schemes are analyzed in section 3. Conclusions are presented in section 4.

II. METHODOLOGY

In this section we describe our simple but highly secure encryption scheme for compressed multimedia distribution (in particular video streaming) over low (Quad-core CPU, 1GB) to mid-power (Octa-core CPU, $\geq$ 2GB) smartphone technologies. Our scheme should be both easily implementable on high-level programming environments such as C and Java (which are available on mobiles systems) and sufficiently fast for securing one-to-one and group real-time video communications. To accomplish this, we propose a new nonintrusive chaos-based encryption methodology that inserts artificial random errors in the VLC bitstream (Huffman or Arithmetic codes) after the codec process. These errors produce valid or invalid codewords with the same or different length (shorter or longer). The effects of errors producing the same codeword-length are local in the bitstream, but they propagate on the decoded data depending on the information carried out by the corrupted codeword (AC, DC, motion vector, header, etc.). The effect of an error producing different codeword-length is more severe; it modifies original codewords bounds, making the reading process incorrect (error propagation) until self-synchronization occurs. The average number of affected codewords during error propagations is called *Mean Error Propagation Length* (MEP*L*), which is an important variable for defining the bit flipping frequency and security of the transmitted data. MEPL has been extensively studied in [31]–[35], and in principle, a symbolic algebraic software is necessary for computing *MEPL*. We follow Takishima *et al.*'s formula [35] for computing MEPL based on crossover probability, which for several numbers of different VLC codes is ∼3-4 codewords.

For the sake of clarity, we briefly describe the scheme in [17], followed by a detailed description of our proposed scheme for improving robustness, performance and elimination of the Wobbling Floating-Point Precision for a perfect deciphering process (independently of the operating system and system designer).

A. OUR PREVIOUS SCHEME

The scheme in [17], consists of three main processes: 1) *random-bit flipping*, 2) *packet division* into L segments and 3) packet *segment shuffling* (Fig.1). These processes depend on chaotic trajectories (or pseudo-random numbers) coming from a floating-point based CML as shown in Eq.1. In particularly, three chaotic trajectories are merged (XOR) to create a new trajectory where only the most significant 27 bits are used for the encryption process (bit flipping and segment shuffling). These actions prevent the attacker from having complete knowledge of the system (we have simplified this process in our new proposal). Once the CML is defined and the entropy coded bitstream is received, the bit flipping operation is applied. The objective is to diffuse and destroy the meaning of the compressed codewords sequence by flipping one bit every $BF = f \cdot Av \cdot MEPL$ bits, where Av is the average size of the entropy codewords in bits (Huffman, Arithmetic, etc.), *MEPL* is defined in codeword units, and $f \geq (1/MEPL)$ is a tunable security factor. The smaller the value of *BF* the higher the bit flipping frequency and corresponding security. Starting from the beginning of the packet, the location of the bit to be flipped is computed as follows:

$$For\ i = 0\ to\ k\ flipBit((random\_number\ mod\ BF) + i^*BF)$$

The bit-flipping location is referenced to the beginning of the packet (i^*BF) in multiples of *BF*, and the bit flipped is within current BF bits (we replaced this mechanism for a more robust one, as we will discuss in the next section). Following the bit flipping process, the packet payload is permuted by an *S-way* shuffling process. Here, the payload is divided into S segments, that represents a security-control variable of the shuffling process. The value of S is randomly changed for

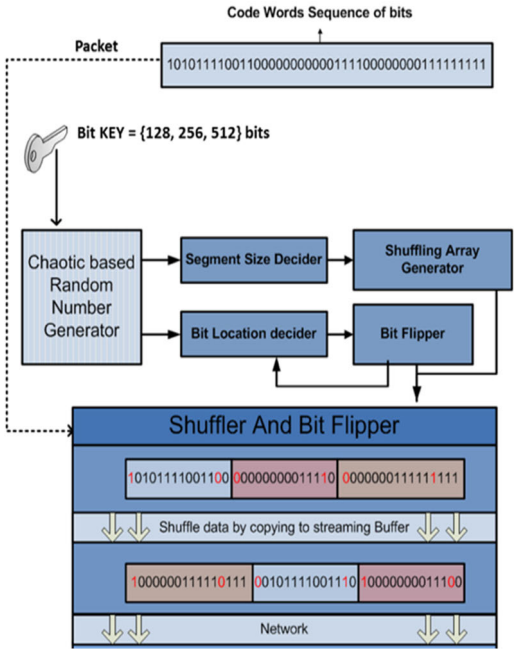


FIGURE 1. Illustration of scheme in [17].

TABLE 1. Difference between Almasalha's scheme [17] and ours.

Encryption Step	Almasalha[17]	Ours
CML	Floating-Point Arithmetic	Integer Arithmetic
Reference for the bit flipping	*Constant*	*2-Step Variable Reference Point:* 1) *Phantom Bit Reference Point (BRP);* 2) *Binary Decision of Bit Flippling: Left or Right wrt BRP*
Size of Trajectories for the Bit Flipping	*[PR=(32,64)/8] Bytes*	*Byte Size (randomly taken)*
Packet Shuffling	*yes*	*No*

every iterated packet depending on the user-defined levels of security; low, medium and high, representing $20 \leq S \leq 35$, $36 \leq S \leq 50$ and $51 \leq S \leq 65$ segments, respectively. The higher the number of segments S in the packet, the higher the security level, since brute force complexity to de-shuffle the packet is $S!$. Once S has been computed, the segment size is $S_g = L/S$ where L is the packet length in bytes.

B. PROPOSED SCHEME

Our new partial encryption scheme is composed of three main processes: 1) a new *integer-based* CML, 2) *phantom DRP*, and 3) a *random byte-trajectory* selection. With these processes, we manage to eliminate several computationally expensive steps in [17] such as, floating-point CML, segment shuffling, truncated trajectories (to 27 bits), random map access, and 3-trajectory XOR computation that affect the overall performance on low-power mobile devices. Table 1 shows the main differences between [17] and our new proposed scheme. A detail description of each step is given next.

1) INTEGER-BASED COUPLED-MAP LATTICE (CML)

Most *continuous* iterative maps exhibit chaotic behavior represented by an infinite periodicity and sensitivity to initial conditions. However, when implemented on digital computers, the state variable takes a finite number of values leading to short limit cycles, non-ergodic trajectories and degraded statistical properties such as invariant probability distribution and correlation [36]. Extensive studies have been conducted to understand and improve the performance of Digital Generators of Chaos (DGC) [37]–[45] and their application to cryptography [46], [47]. Several techniques have been utilized to extend the periodic orbit length of DGC, among the most important for the present work and of primary interest in the field of chaotic cryptography is CML. CMLs are constructed by an N-network of dynamically evolving maps interacting through some coupling rule over a neighborhood:

$$\begin{aligned} X_{i,j} &= (1-\epsilon) f\left(X_{i,j-1}\right) \\ &\quad + \epsilon H\left(X_{i,j-1}, \ldots, X_{N,j-1}\right) \\ H\left(X_{i,j-1}, \ldots, X_{N,j-1}\right) &= \sum_{i=1}^{N} w_i X_{i,j-1} \end{aligned} \tag{1}$$

where $X_{i,j}$ is the chaotic variable (or random trajectory) of the $i^{th}(\leq N)$ map at state or iteration $j > 0$, $f\left(X_{i,j-1}\right)$ represents each individual map (local term) and H the coupling function (linear/nonlinear interaction term) with weights w_i, such that $\sum_{i=1}^{N} w_i = 1$. By varying the strength of the control parameter ε different phases may appear, such as *localized* chaos or *spatio-temporal intermittency* chaos [44]. That is, when the weight of the coupling is weak, the system can be regarded as a local map perturbed by contributions from other sites, thus maintaining its main individual properties. On the other hand, when the weight of the coupling is large, the system reaches an asymptotic collective behavior characterized by intermittent periodic chaotic cycles. There are several attractive characteristics of CML from the cryptographic point of view: a) the chaotic regime appears sooner than a single map [44]; b) extended cycle length; and c) robustness to attacks. Regarding the coupling function H, two variants have been used for studying the dynamics of CML, l*ocal* and *global* coupling. The latter, considers the effects of the nearest neighbors of a given lattice site, while the former considers the interaction

of each map with the "mean field" generated by all lattice sites.

Our proposed pseudo-random number generator to be used in the bit-flipping and random byte-trajectory selection processes is based on *global coupling* and *integer arithmetic*, which can be written in generalized form as:

$$\begin{aligned} X_{i,j} &= (1-\varepsilon_n) \odot f\left(X_{i,j-1}\right) \\ &\quad + \varepsilon_n \odot H\left(X_{i,j-1}, \ldots, X_{N,j-1}\right) \\ H\left(X_{i,j-1}, \ldots, X_{N,j-1}\right) &= \odot_{i=1}^{N} w_i X_{i,j-1} \end{aligned} \tag{2}$$

where $N \geq 6$ is the number of maps, $\odot$ represents a logical (XOR $\rightarrow \oplus$, OR $\rightarrow |$, AND $\rightarrow$ &, left-shift $\rightarrow \ll$, right-shift $\rightarrow \gg$, etc.) or arithmetic operator, or a combination of operators. For the purpose of this work, our final CML is defined as follows:

$$\begin{aligned} X_{i,j} &= f\left(X_{i,j-1}\right) \\ &\quad + \varepsilon_n \& H\left(X_{i,j-1}, \ldots, X_{N,j-1}\right) \\ H\left(X_{i,j-1}, \ldots, X_{N,j-1}\right) &= \oplus_{i=1}^{N} X_{i,j-1} \end{aligned} \tag{3}$$

where $\varepsilon_n = 2^n - 1, 16 \geq n \geq 0$ represents a 16-bit random integer number that can be fixed or dynamically changed on iteration basis (increasing security), H is a global coupling function based on XOR operation over the N previous chaotic variables. ε_n, adds the first n-bits of the global coupling function H, to the corresponding local map $f(X_{i,j})$ represented by the digitized Rényi Map defined as [48]:

$$\begin{aligned} X_j &= f\left(X_{j-1}\right) = \left(b \cdot X_{j-1} + \left\lfloor \frac{X_{j-1}}{2^m} \right\rfloor\right) mod 2^{PR} \\ (X, m) &\in \left\{1, 2, \ldots, 2^{PR} - 1\right\} \end{aligned} \tag{4}$$

where $b \epsilon \mathbb{Z}^{>0}$ is the control parameter and PR is the CPU bit-precision (32 or 64 bits).

It is important to point out that H can be modified to include previous plaintext/ciphertext values to induce diffusion in case of attacks, generating a completely different ciphertext with respect to the original output. Plaintext/ciphertext feedback may be introduced as follows:

$$H\left(X_{i,j-1,\ldots,}X_{N,j-1,}P_{ant}\right) = \oplus_{i=1}^{N}\left(X_{i,j-1}\right) + PC_{ant} \tag{5}$$

The number of maps $N(\geq 6)$ depends on the size of the system-key K. 10 bytes are used for the initialization of each map, where the first 4 bytes initialize the chaotic variable X, 4 additional bytes to parameter b, and 2 bytes for m. For $N = 6$, the minimum K must have $B = 480$ bits of length. It is possible though, that the length of the system key be $256 \leq B < 480$ bits; in this case, we generate a CML with $n < 6$ maps and use the output to initialize the rest of the variables and parameters (including ε) until all 6 maps are created. The N PR-bit trajectories produced at each iteration j in Eq.3, are used in a byte-to-byte basis for the *DRP* and bit flipping (the idea is to get the most out of every trajectory). Each iteration of Eq.4 produces $Tr = N \cdot (PR/8)$ byte trajectories, corresponding to $Tr/2$ flipped bits in the bitstream (2 bytes are used for flipping one bit). {32-64}-bit trajectories are sequentially stored in memory and randomly retrieved at a byte level, as shown in Fig.2 for a 2-map *CML* example (discussed in the next section).

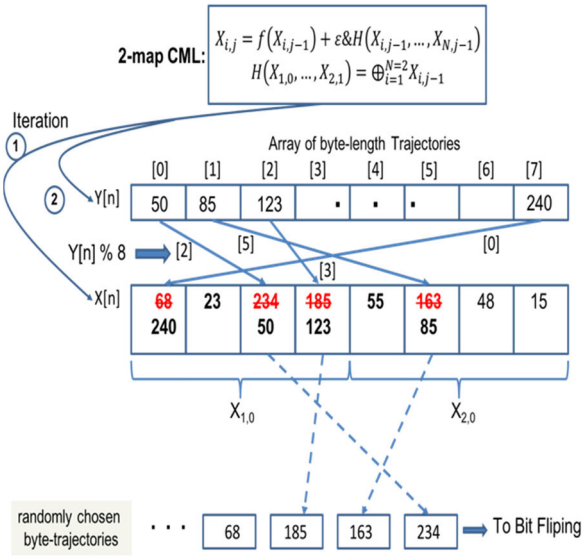


FIGURE 2. Random byte extraction from PR-bit trajectories $X_{i,j}$.

In order to increase the sensitivity of the encryption system to system-key changes, Eq.4 is randomly iterated $20 \leq RT \leq 50$ times and the N-map output becomes the initial state in the encryption system. According to our experiments, $RT = 20$ represents the minimum number of iterations required for a perturbed chaotic trajectory to diverge from its original trajectory when the magnitude of the perturbation is 1 (minimum magnitude), guaranteeing that a bit change in K, will affect the entire system output or ciphertext.

2) BIT-FLIPPING

This step is aimed at destroying the meaning and synchrony of the entropy coded bitstream. The Huffman code tends to recover from errors after a certain number of codewords, unlike Arithmetic code which exhibits poor resynchronization capabilities [5], [30]. Consequently, the frequency of the bit flipping depends on the selected VLC code and corresponding MEPL value. Random errors in the bitstream are induced and propagated in such a way that becomes impossible for an attacker to resynchronize back to the original sequence, unless he finds out the system-key K. This is an effective method to hide partial information. However, scheme in [17] overexposes the location of the bits flipped and related CML pseudo-random numbers under a known/chosen attack. The reason is that, every bit flipped is selected with respect to a fixed reference point along the packet, which happens to be a multiple of *BF* (knowing *BF* discloses the random bit flipping number). We extend the bit-flipping process to make it more robust against known/chosen attacks by considering a phantom or invisible *DRP*. The *DRP* is not exposed to attacker and does not reveal information

about the random numbers involved in the bit-flipping (its effect on security is discussed in section 3). The scheme consists of the following steps:

1) Define the mean average propagation-length in codeword units *MEPL*, the level of security $f \geq 1$ (lower values represent higher bit flipping frequency), the average size of the entropy codewords *Av* in bits and compute the bit flipping block length $8 \leq BF = f \cdot Av \cdot MEPL \leq 128$.
2) For every CML iteration (Eq.3), organize the N *PR*-bit output trajectories $X_{i,j}$, $1 \leq i \leq N, j \geq 0$ into an array of bytes $0 \leq X[n] \leq 255$ of length LEN $= N * (PR/8)$ (as in Fig.2).
3) Do until the end of input multimedia data: *initialize* $n = DRP = rand = 0$, iteration $= 1$:
   a. Compute the next N *PR*-bit trajectories ($Y_{i,j} = X_{i,j+1}, 1 \leq i \leq N$) into an array $Y[n]$ of *LEN* bytes (similar to $X[n]$).
   b. Use $Y[n]$ to randomly select trajectories in $X[n]$ as follows: Compute $rand = Y[n] \, mod(N*(PR/8))$ (see Fig.2), a random number index pointing to a byte location in X[n].
   c. X[rand] is used for the generation of the new bit reference point, located at:

$$DRP(BF, X) = (pDRP + X[rand]) \ \ modA$$

   where $A = (iteration * BF - 2)$, *pDRP* is the previous *DRP* (initially zero). *DRP* is bounded above by the variable *A*, whose value depends on the *iteration* counter (number of bits flipped) and the predefined constant BF. Probabilistically speaking, this strategy attempts to spread out the bit-flipping process evenly along the bitstream.
   d. Replace $X[rand]$ with the next element n of $Y[n]$. Once $Y[n]$ $(1 \leq n \leq LEN)$ has been fully iterated, it is renewed with the next N-CML trajectories to continuously feeding $X[n]$. Random selection of byte-trajectories does not overexpose full trajectories coming from one map; any trajectory found by the attacker may now come from one, two, . . ., or N different maps, increasing the security of the system.
   e. Once *DRP* is defined, a decision is made whether the bit to be flipped will be at the right or left side of *DRP*. If *DRP* is even (odd), the encrypted bit will be at the left (right) side of *DRP*. The bit to be flipped is calculated as:

      Right side (R):
      i. $BitflipPos = pDPR + (X[next\_rand] \, mod \, \lfloor iteration^{*}BF \rfloor)$; where the module operator ensures the flipped bit is between [*DRP, iteration*BF*].
      ii. $pDPR = BitflipPos$;

      Left side (L):
      i. $DPR - [X[next\_rand] \, mod \, (DRP - pDRP)]$; where the module operator ensures the flipped bit is between (*pDRP,DRP*).
      ii. $pDPR = DPR$;

TABLE 2. Smartphone specifications.

Smart Phone Model	Chipset	CPU	Memory (Gb)	Android OS
Moto G5+	Qualcom MSM8953 Snapdragon 625	Octa-core 2.0 GHz Cortex-A53	2	7.0
Moto G4+	Qualcom MSM8952 Snapdragon 617	Octa-core 4x1.5 GHz 4x1.2 GHz Cortex-A53	2	7.0
Samsung Galaxy Grand Prime +	Mediatek MT6737T	Quad-core 1.4 GHz Cortex-A53	2	6.0.1
HTC Desire 650	Qualcom MSM8928 Snapdragon 400	Quad-core 1.6 GHz Cortex-A7	2	6.0.1

TABLE 3. CML initial conditions for 256-bit system-key.

Map Number	Initial Value $X_{i,j}$	Parameter b
1	0x87345377	513
2	0x3134C07B	2049
3	0x5331F57D	17
4	0x4b030A971	257
5	0xA7B3C	23584
6	0x28463	9821721
ε = 65356, j = 5		

Fig.3, shows an example of the bit flipping steps for $BF = 10$ bits. The first random *DRP* falls in the bit number 6 (even number) out of $BF = 10$, therefore one of the bits at the left side of *DRP* will be flipped. In this case, the bit flipped corresponds to bit number 3 and sets $pDRP = 6$. In the second iteration, the new *DRP* can go from bit locations ($pDRP = 6$, $2^{*}BF = 20$), randomly falling in $DRP = 11$ (odd number), meaning that one of the bits at the right side of *DRP* will be flipped. In this case the random bit flipped corresponds to bit 13, and the new reference point for the next iteration becomes $pDRP = 13$, the position of the bit flipped. If the flipped bit is at the right side of *DRP*, then *pDRP* is equal to the *position of the last bit flipped* (see *3e* step in algorithm above).

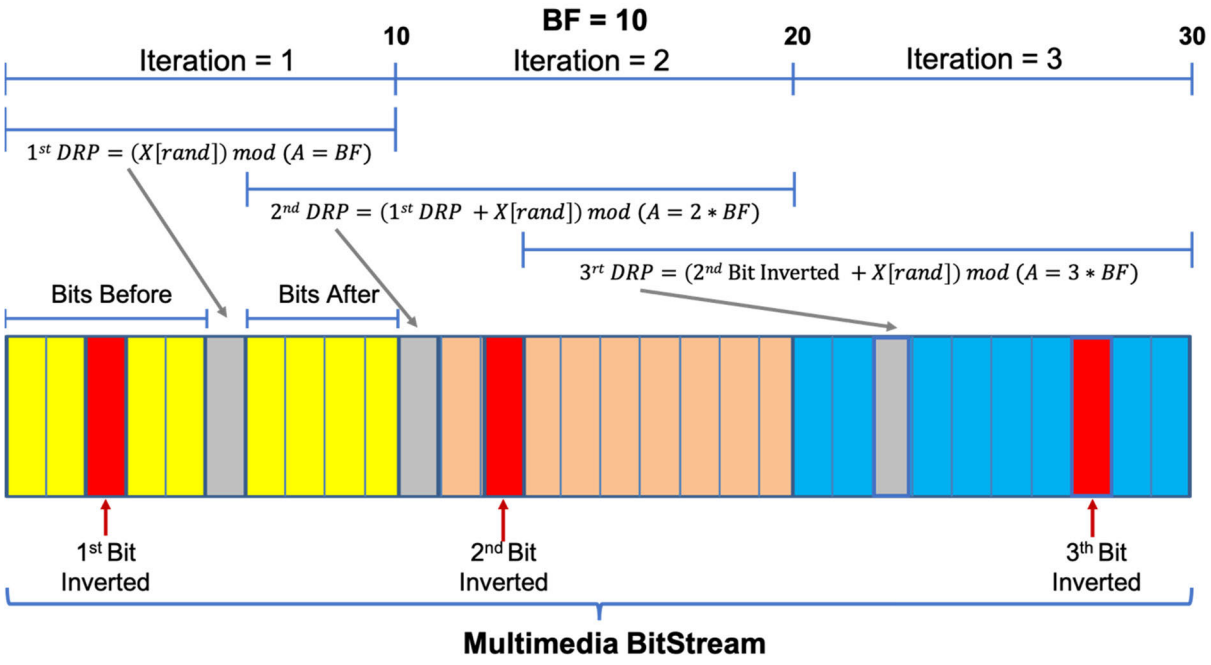


FIGURE 3. Bit flipping strategy wherein bits are flipped (to the left or right) with respect to a variable reference point (BPR).

As can be seen, the range of DRP ($pDRP \leq DRP \leq iteration * BF$) is variable along the encryption process; $iteration^*BF$ is used as an upper boundary for the bit flipping, and not as a measure of bit flipping frequency anymore. There is a probabilistic relationship though between bit flipping frequency and BF. Once a bit has been flipped, the next $iteration^*BF$ block may overlap with previous block, therefore more than 1-bit is likely to be flipped every $iteration^*BF$ bits. Bit frequency (per packet) oscillates probabilistically between L/BF and $L/2$ (where L is the packet-length in bits), the minimum and maximum bit flipping frequencies respectively. BF is inversely related "*probabilistically speaking*" to bit-flipping frequency, the smaller the BF (or f) the higher the bit flipping frequency.

III. RESULTS

The performance of the proposed scheme is evaluated for different gammas of smartphones architectures (Table 2) and programming languages (C and Java) under Android-OS. The CML set up is according to Table 3. The packet or bitstream length considered is 400 bytes with average bit flipping frequencies of 1/32, 1/64, and 1/128 (1 bit randomly flipped every 32, 64 or 128 bits respectively). The following test are taken in consideration for the overall performance of the scheme: a) Ciphertext (compressed video) and CML sensitivity to initial conditions; b) Security Analysis; and c) Scheme performance. The deciphering process performs the same operations as the cipher, therefore both have the same time complexity.

A. CIPHERTEXT AND CML SENSITIVITY TO INITIAL CONDITIONS

In previous section, we pointed out the relevance of flipping at least 1 bit every MEPL codewords to destroy any possible resynchronization of the entropy coded bitstream. *Is this sufficient from the security point of view?* Before we answer this question (see section 3.2), let us first visualize the effect of a bit inversion on compressed video data. Under random errors in the coded bitstream, data visualization is not always possible because codewords may be invalid, terminating the decoding process. Precautions were taken for inserting bit errors on DC or AC coefficients to generate valid codewords and continue decoding. The inversion of only one bit, along with error propagation affects dramatically the quality of the decoded data as shown in Fig.4b. At a higher error rates (or higher bit flipping frequencies in our case), the effect of our partial encryption scheme completely transforms compressed data into a non-decodable noisy look pattern (Fig.4c). In the case of attack, our SE force the attacker to either guess the system-key, guess the flipped bits (using original codewords to guess the error bit), or break the CML system through a known/chosen plaintext/ciphertext attack. As we will see next, the lowest complexity attack for breaking our scheme is through the system-key. Another important behavior of the encrypted data at a high bit flipping frequency is that, the output (ciphertext) histogram is uniform and independent of the input histogram shape (Fig.4d).

Important properties of chaos-based encryption systems are their sensitivity to initial conditions, randomness, and unpredictability, which are directly related to our proposed

TABLE 4. NIST test suite results from randomness evaluation of the proposed integer-based CML.

Test	p_{value}	$p_{\text{value}} \geq \alpha = 0.001$
Approximate Entropy	*0.401440*	*Pass*
Block Frecuency	*0.272586*	*Pass*
Cumulative Sums	*Forward test: 0.995925, Reverse test 0.996246*	*Pass*
FFT	*0.961969*	*Pass*
Frequency	*0.990787*	*Pass*
Linear Complexity	*0.522735*	*Pass*
Longest Run	*0.945150*	*Pass*
No overlapping Template	*Accepted p_{values}: 148 out of 148*	*Pass*
Overlapping Template	*0.656608*	*Pass*
Random Excursions	*8 out of 8*	*Pass*
Random Excursions Variants	*18 out of 18*	*Pass*
Rank	*0.223181*	*Pass*
Runs	*0.445310*	*Pass*
Serial	*$p_{values1}$:0.130233, $p_{values2}$: 0.703281*	*Pass*
Universal	*0.098671*	*Pass*

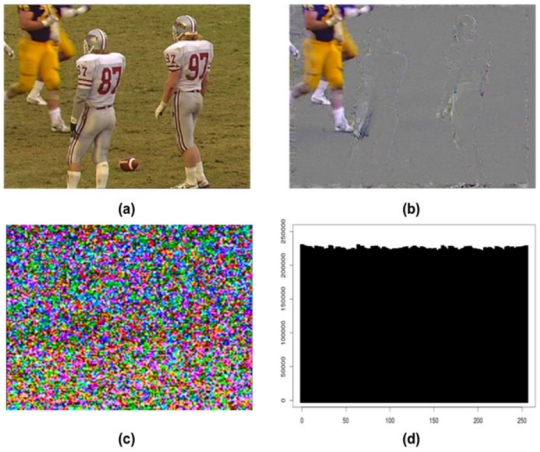


FIGURE 4. (a) Original football video frame, (b) effect of a random 1 bit flipping, (c) encrypted frame output (ciphertext), and (d) ciphertext histogram.

integer-based CML. We will prove these properties using a statistical package developed by the National Institute of Standard and Technology (NIST) [49]. The NIST test suite consists of 15 tests formulated under the null hypothesis (H_0) that a sequence (of the order 10^3 to 10^7 elements) is random. The tests are based on a specified $0.01 \leq \alpha \leq 0.001$

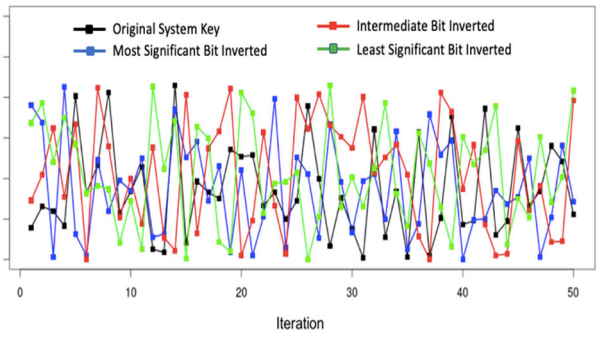


FIGURE 5. Sensitivity to system-key changes by inverting only 1-bit. Original sequence is represented in black, most significant, intermediate, and least significant bit inverted in blue, red, and green color respectively.

value, representing the probability that the sequence is not random when it really is random, and a *P-value* representing the strength of the evidence against the null hypothesis respectively. If *P-value* $\geq \alpha$, then the sequence appears to be random. We performed over 200 different tests on Eqs.3 and 4 for $N = 6$, with random initializations of the state variable X, parameter b, and $5 \leq j \leq 10$, for fixed $\varepsilon = 255$. All sequences individually passed each one of the 15 tests with different strength; the average output is shown in Table 4. We found that $5 \leq j \leq 10$ is the best range for producing chaotic sequences, for $j < 5$, results were not adequate. Now, *is Eq.3 sensitive to initial conditions*? In Fig.5 we show the sensitivity of the CML to system-key changes, where only one of the N maps is plotted for clarity. The sequence with original system-key K is represented in black, most significant bit inverted in blue, least significant bit inverted in green, and intermediate bit inverted in red. As can be seen, all 4 trajectories diverge from the very beginning up to the end of the process, ensuring that a system-key attack (section 3.2) will produce a totally different ciphertext. Same result is obtained when the modification (or attack) happens in map's parameters (b, j), intermediate trajectories (changing one bit in the i^{th} map trajectory), coupling function H, or ε. In conclusion, the proposed CML is excellent as a PRNG with extreme sensitive to initial conditions. A more visual example is shown in Fig.6, where the ciphertext (Fig.6b) is deciphered with the wrong system-key (least significant bit inverted) yielding Fig.6c, the decoded image is not identifiable.

B. SECURITY ANALYSIS

We now discuss the order of magnitude required for breaking our code using both bit flipping brute force attack and known/chosen attack. System-key brute force attack is related to its length in bits, with a user-defined minimum complexity of $2^{B \geq 250}$ (for $N = 6$, System-key attack is 2^{480}).

1) BIT-FLIPPING BRUTE FORCE ATTACK

Attacker wants to find out the bits flipped in the bitstream. As mentioned in section 2.2.2, our bit-flipping process involves a random bit-reference point $0 < DRP \leq 2^{k=8}$, selected from a variable-length window shifted along the

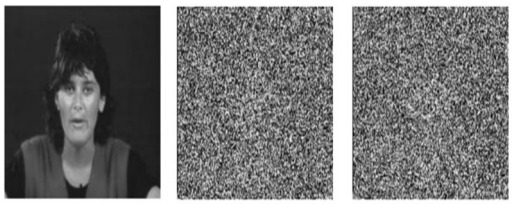


FIGURE 6. (a) Original image, (b) Encrypted image, and (c) Decrypted with wrong system-key (least significant bit changed).

TABLE 5. Comparson of our proposed and Almasalha's schemes Complexity Attack.

Type of Attack	Bit Flipping per packet (N = 6, BF=18, PR = 32 bits, E[NB] = 114)	
	Almasalha[17]	Our Scheme
Brute Force (Bit-Flipping)	$2^{k(NBF)} = 2^{1068}$	$BF \cdot 2^{2k(NBF)} = 2^{2136}$
Length of Trajectories for the Bit-Flipping	$(2^{2N-PR+k}) = 2^{389}$	$BF \cdot E(NB)! \cdot 2^{8 \cdot E(NB)} = 2^{1537}$

bitstream (see Fig.3). Once *DRP* is defined, the bit flipping position is another random number $R \leq 2^{k=8}$ to either left or right side of *DRP*. Altogether, *DRP* and *R* have an average complexity of 2^{2k}. The total Number of Bit Flipping (*NBF*) in a packet of length *L* bits becomes a random variable as well, between $L/128 \leq NBF \leq L/8$, for which the complexity of brute force attack per packet can be expressed as:

$$O\,(bit\ flipping) = BF \cdot [DRP \cdot R]^{NBF} = BF \cdot 2^{2k \cdot NBF} \quad (6)$$

where *BF* is fixed in the entire encryption process. For *P* total number of packets, we have:

$$O\,(bit\ flipping) = \mathrm{BF} \cdot 2^{\mathrm{P} \cdot 2\mathrm{k} \cdot \mathrm{NBF}} \quad (7)$$

The security is increased by a power of two with respect to [17], as shown in Table 5. The security provided by this scheme is tunable, it can be modified by increasing or decreasing the bit flipping period (BF) or frequency (1/BF) through the value of f in $BF = f \cdot Av \cdot MEPL$ for $f \geq 1$. This result can be used for answering our question (see previous section) about how much security is provided by *flipping at least one bit every MEPL codewords.* Assuming for the moment that one bit is flipped every *BF* bits (not true in our case, since 1/*BF* underestimates our real number of bits flipped in a packet), then for $f = 1$, $Av = 6$ bits, and $MEPL = 3$ codewords (following [35]) we get $BF = 18$ bits. For a packet length of *400 bytes*, average $k = 6$ *(for representing BF)*, $NBF = L/BF = 178$, the order of the attacks becomes $\sim 2^{2136}$ (just for 1-packet), which is highly secure. It is possible to increase the security (if desired) to our maximum allowable value of one bit flipped every $BF = 8$ bits, to get a minimum complexity attack of $\sim 2^{4800}$. Security can also be decreased to users' needs proportionally to *NBF*; in our case, we consider $\sim 2^{300}$ as the lower security limit corresponding to $NBF = 25$ bits ($BF = 128$ bits) per packet length of 400 bytes. For an I-Frame of size 2KB (320 x 240 coded in MPEG4 format) the complexity of the attack with the lowest (maximum) security would be $\sim 2^{1500}$ ($\sim 2^{24000}$), representing $\sim$ 5 packets attack.

2) KNOWN/CHOSEN PLAINTEXT ATTACK

In the case of known/chosen plaintext attacks, the target is the CML through vulnerabilities in the bit flipping (Eqs.3 and 4). Even though bits flipped are vulnerable to this attack (their position is easily revealed), it is not easy to break through the CML because of the unknown *DRP* and the random selection of byte-trajectories. Recall that *DRP* is computed as (section 2.2.2):

$$DRP(BF, X) = (pDRP + X\,[rand])\ mod\ A$$

Assuming the attacker knows the inverted bits in the packet, the aim at first is to find the byte trajectories $X[n]$ and *BF*. The complexity of the first bit flipped in the packet is:

$$BF \cdot 2^{k} = 2^{2k}$$

where, 2^{2k} represents the join complexity of *BF* and the byte-length trajectory ($X[n]$) for deciding if a bit is flipped to the right or left side of *DRP*. The attacker needs to find $2N$ consecutive PR-bit trajectories (Eq.3) and solve for the N parameters $1 \leq b \leq 2^{PR=\{32,64\}}$ and $m \in \{1, 2, \ldots, PR-1\}$ in the Rényi map (Eq.4). Assuming ε and H are known, the complexity can be represented by:

$$2^{2N \cdot PR} \quad (8)$$

The effect of random byte selection is an additional complexity that needs to be reflected into Eq.8. Before this happens, the attacker must expose two consecutive sets of N trajectories and solve for b and m. The easiest way is to attack the system from the very first iteration, in which the attacker knows that the first $NB = N * PR/8$ byte-trajectories may come from the first and/or second iterations of the CML (X[n] and Y[n]). After this iteration things complicates out, trajectories may now come from $i \leq N$ different maps and $j \leq IterationNumber$ different iteration times (see Fig.2), which is a major problem since the goal is to extract bytes from two consecutive iterations. From the second iteration on, the attacker needs to figure out how many bit-flipping are needed until all byte-trajectories in Y[n] appear again; this is to ensure that the attacker will have on hands two consecutive set of N trajectories for breaking out the system. The aim is bringing together in perfect order these $2N$ trajectories to recover original PR-bit trajectories. Following the above steps, we randomly draw *NB* bytes from X[n] corresponding to the first iteration for the bit flipping and replace them with bytes in Y[n]. For the second iteration, drawn bytes may come from the first iteration (original X[n] values) or from second iteration (Y[n]), and son so forth for the third, fourth, etc. iterations. The attacker is waiting for the all *NB* bytes in Y[n] to be called out (only those belonging to the second iteration), which is related to the expected number of random calls to

empty Y[n], represented by:

$$E\,(NB) = 1 + \frac{NB}{NB-1} + \frac{NB}{NB-2} + \ldots + NB = NB\sum_{i=1}^{NB}\frac{1}{i} \tag{9}$$

This is similar to the *NB*-face die problem stating "how many times must a *NB*-sided die be rolled out until all sides appear at least once". We initiate by randomly calling the first byte from X[n], then there are *NB-1* different random numbers we could call in, taking on average $1/((NB-1)/NB) = NB/(NB-1)$ calls to get a different n from X[n]. Third random call requires $NB/(NB-2)$, and the process continues until all *NB* calls are completed. For $PR = 32$ bits, the total expected number of calls (including the 24 bytes drawn from the first iteration) is $E(NB) + NB = 90 + 24 = 114$. So, the attacker needs to track down 114 random calls (on average) to find every byte belonging to the first two consecutive iterations of *2N*-CML trajectories. The good news for the attacker is that he only needs one packet to attack the system (first packet); the bad news is the complexity of the attack itself, which involves an additional permutation process of $2N(PR/8)$ bytes in order to find the right trajectories. The final complexity of known/chosen plaintext attack can be represented by:

$$\begin{aligned} O\,(CML\ Attack) &= BF \cdot E\,(NB)! \cdot 2^{8\cdot E(NB)} \\ &= BF \cdot (114!) \cdot 2^{912} \\ &= 2^{912+(k=6)+619} = 2^{1537} \end{aligned} \tag{10}$$

Again, a much better option is to attack the system-key. Note that, the complexity of known/chosen attack (and thus security of the system) *is not* directly proportional (probabilistically speaking) to the bit flipping frequency *NBF*, as in brute force attack. Therefore, it is possible to use a wide range of bit flipping frequencies without affecting the security of the system. For very low frequencies, the limit is the brute force attack. Known/Chosen security attack can be modified by increasing the number of maps N in the CML (Eq.3), and/or using more bytes in X[n] and Y[n] (such as unsigned short or unsigned int) for the computation of all random numbers involved in the encryption.

C. SCHEME PERFORMANCE

We describe now the encryption speed of our proposed scheme on low to mid-power smartphones under Android OS (see Table 2). Experimental tests show that our encryption scheme performs differently according to the arithmetic involved in the computation (integer or floating-point arithmetic), CPU type, version of the operating system, and programming language (Java and C).

1) FLOATING-POINT VS INTEGER ARITHMETIC

The first experiment evaluates the performance gain of our new integer-based CML (I-CML) against the floating-point based CML (F-CML). The timespan for evaluating five-hundred thousand iterations were recorded for two different

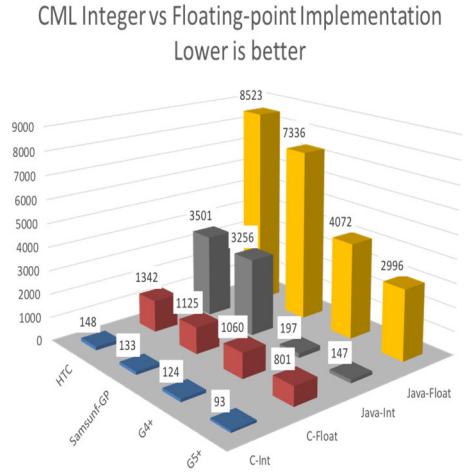


FIGURE 7. Integer and floating-point CML processing time using C and Java programming languages.

TABLE 6. Recommended bit flipping frequencies for secure video transmission.

Security Level	Bit Flipping Frequency Range
High (HF)	$\frac{1}{8} \geq HF \geq \frac{1}{32}$
Medium (MF)	$\frac{1}{32} \geq MF \geq \frac{1}{64}$
Low (LF)	$\frac{1}{64} \geq LF \geq \frac{1}{128}$

programming environments, Java under Android RunTime-ART and C under Native Developing Kit-NDK (see Fig.7). On low-power smartphones under Marshmallow (HTC and Samsung-GP), I-CML C implementation came out to be ∼9 times faster than corresponding C F-CML, whereas Java I-CML is only twice as faster than Java F-CML. On mid-power smartphones under Nougat (G4+ and G5+) same gain is maintained in C (∼9), while a deep gap is found in Java where I-CML reached as high as ∼21 times faster than corresponding Java F-CML implementation (a possible explanation of this behavior is discussed in the next section). Cross-comparison of C vs Java, C I-CML is on average 46 and 14 times faster than corresponding Java F-CML and Java I-CML respectively. The migration from floating-point to integer arithmetic on smartphones, provides significant performance gain without degrading the chaotic properties of the CML as discussed in section 3.1.

2) PERFORMANCE OF THE PROPOSED INTEGER-BASED ENCRYPTION SCHEME

The next set of experiments evaluate our integer-based encryption scheme implemented on both C and Java. The experiments consider *High (HF), Medium (MF),* and *Low*

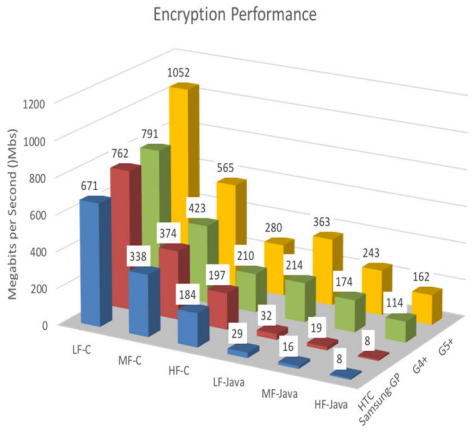


FIGURE 8. Encryption performance under C and Java for different bit flipping frequencies (HF, MF, and LF) and smartphone platforms.

(LF) bit-flipping frequencies, which may also be considered as "*security levels*" with respect to brute-force attack (the higher the frequency the higher the complexity of the attack). In the case of known/chosen plaintext attack the story is a bit different and beneficial to our proposed scheme, the complexity of the attack is independent of the bit flipping frequency which can be reduced without sacrificing system security (see section 3.2.2). Something to keep in mind when using low bit flipping frequencies (lower than 1 bit flipped every 128 bits) is the resynchronization capability of entropy coders, which may reveal (with very low probability) sporadic blocks of information depending on the affected portion of the compressed video (transform coefficients, motion vectors, etc.). Table 6, indicates the recommended security levels in terms of the bit flipping frequency, e.g. *HF* is considered when the bit flipping frequency is between $1/32 \leq HF \leq 1/8$, that is one bit is flipped at least every 32 bits and at the most every 8 bits (this is on average, since variable length block *DRP* is used). In our experiments, we use the lowest frequency at each security range, that is 1/32, 1/64, and 1/128 for *HF*, *MF*, and *LF* respectively. In particular, the bit flipping frequency of 1/32, corresponds to Takishima's et.al. [35] general entropy coding recommendation to avoid natural bitstream resynchronization (*MEPL∼3-4* codewords with average codeword length of *Av∼8* bits). This recommendation can be loosened up according to user application's needs and/or CPU strength; we setup our lowest frequency limit to 1/128, corresponding to a minimum accepted brute force security of 2^{250}.

In general, high-performance behavior is practically seen on both C and Java for all bit flipping frequencies and smartphone platforms, the only exception is Java on Low-range Smartphones (L-SM) under Marshmallow (HTC and Samsung-GP) as shown in Fig.8. Average speeds are overwhelming, with an overall average (along all bit flipping frequencies and smartphone platforms) of $C = 544$ *Mbs and* $Java = 115$ Mbs, for a total $C/Java$ ratio of 4.73 (C is on the average four times faster than Java). In particular, C offers excellent cross-platform average (along smartphones) of

TABLE 7. Minimum and recommended download and upload speeds for Skype video calls [51].

Call Type	*Minimum Download/Upload speed*	Recommended *Download/Upload speed*
Calling	*300kbps/300kbps*	*100kbps/100kbps*
Video Calling / Screen Sharing	*128 kbps/128 kbps*	*300kbps/300kbps*
Video Calling (high-quality)	*400 kbps/400 kbps*	*500kbps/500kbps*
Video Calling (HD)	*1.2Mbs/1.2Mbs*	*1.5Mbs/1.5Mbs*
Group video (3 people)	*512 kbps/512 kbps*	*2Mbs/512kbs*
Group video (5 people)	*2Mbs/128kbs*	*4Mbps/512kbps*
Group video (7+ people)	*4Mbps/128kbps*	*8Mbps/512kbps*

TABLE 8. Protection provided for recommended data volume range of encryption between 0.7% - 3% ($32 \leq BF \leq 128$) and $N = 6$ chaotic maps (Eq.3).

Type of Attack	Security Provided
System-Key	2^{480} increases with N
Bit Flipping Attack (Eq.7)	$2^{400} - 2^{1600}$ Per packet
Known/Chosen Plaintext Attack (Eq.10)	2^{1537} (32bit CPU) 2^{3836} (64bit CPU)

819Mbs, 425Mbs, and 218Mbs for LF, MF, and HF respectively. With variable CPU-power capabilities, C continues offering excellent performance with cross-frequency average (along bit-flipping frequencies) of 421Mbs (min = 184, max = 762) on L-SM (HTC and Samsung-GP) and 554Mbs (min = 210, max = 1052) on M-SM (G4+ and G5+), for a ratio $C_{M-SM}/C_{L-SM} = 1.31$ (31% M-SM performance gain). The maximum encryption speed for C is 1Gbs (Gigabits per second) corresponding to LF on the G5+ smartphone.

Java performance is drastically distinct between L-SM and M-SM. On L-SM running Marshmallow, Java achieves an unbelievable low performance average of 19Mbs (although sufficient for real-time video communications), while surprisingly on M-SM running Nougat there is a clear turnover, the average climbs up to 212Mbs (min = 114, max = 363) for a 1015% performance gain. A possible explanation to this behavior may be attributed to Java improvements due to

TABLE 9. Comparison of the encrypted data volumes of several selective encryption schemes.

Partial Encryption Schemes	Encrypted Information	Bitrate Increase	Codec Independent	Scalable (Tunable Encryption)	Encrypted Data Volume	Brute Force Attack Complexity (Encryption Domain)	Average Complexity Overhead
Fawas, et.al. [50]	JPEG 2000 Bitstream: Transform Coefficient Substitution and Diffusioin	No	Intrusive	No	4%	2^{128} (AES-128)	43%
Massoudi, et.al. [10]	JPEG 2000: High and Low Resolution Subband	No	Intrusive	Yes	5.4% (to guarantee data security)	2^{128} (AES-128)	613% (source [50])
Lian & Chen [26]	Image Wavelet Coefficients at Different Frequencies	Yes	Intrusive	Yes	10%	Subjective Perceived Security (AES)	8.9%
Wang, et.al. [46]	H.264/AVC Intra prediction mode and Motion Vector difference bitstream, Quantization Coefficients	Yes	Intrusive	No	6.7%	2^{128} (AES-128)	~60%
ELIMINAR Shahid, et.al. [8]	Encoded Btstream-H.264/AVC CAVLC and CABAC	No	Intrusive	No	7%-29% 18% (average)	2^{128} (AES-128)	0.5%-4%
Chung, et.al. [55]	H.264 AC/DC Coefficients and Motion Vectors	Yes	Intrusive	Yes	1.5%-15%	2^{128} (AES-128)	Not Reported
Hamidouche, et.al. [54] Complexity overhead <6%	HEVC extension CABAC Bitstream	No	Intrusive	Yes	~17%	Not Reported (Chaotic)	≤ 6%
Proposed Scheme	Encoded Bitstream	No	Non-intrusive	Yes	0.7% - 3% (Recommended)	$2^{300} - 2^{1200}$ (Up to 2^{4800} at BF=8)	**JPEG2000:** 1.3%-3.5% **H264:** 0.44%-2.4%

Just-In-Time (JIT) compiler introduced in Android 7.0, JIT complements RunTime-ART's current ahead-of-time (AOT) compiler, improves runtime performance, saves storage space, and speeds up applications (for more information see: https://source.android.com/devices/tech/dalvik/jit-compiler). Comparing C vs Java, C outperforms Java by 2100% ($C_{L-SM}/Java_{L-SM} = 22$) on L-SM, while on M-SM the performance difference is considerably reduced to only 161%

TABLE 10. i5 platform specifications for performance comparison between our scheme and Homidouche *et al.* [54].

CPU Specifications	Intel i5-460M (our configuration)	Intel i5-4300M Processor (Hamidouche, et.al, [54])
Total Cores	2	2
Total Threads	4	4
Base Frequency	2.53 Ghz	2.6GHz, Intel Boost 3.3 GHz
Bus Speed	2.5 GT/s	5 GT/s
Memory (RAM)	Max 8GB DDR3	Max 32Gb DDR3L 1333/1600
Launch Date	Q3'10	Q4'13
OS	Ubuntu 20.04.3, 64-bit	Ubuntu 14.04, 64-bit

$(C_{L-SM}/Java_{L-SM} = 2.61)$; the performance gap between C and Java is less apparent now. Although, on low-power smartphones with 32-bit CPU and Android 6.0 or lower, our standard C encryption implementation is recommended.

With the above encryption performance, secure video calls on smartphones can be handled easily independently of the video codec technology (MPEG4, H.264, etc.), resolution and #frames/sec. Videoconferencing applications such as Skype has different bandwidth requirements depending on the quality and number of participants in the video call. Download bandwidth for one-to-one to group (7+ participants) video calls may go from 0.5-8.0 Mbs (Table 7) [50], which represents 0.2-3.7%, 0.12-1.8%, and 0.06-0.9% of the average processing speed in C for *HF*, *MF, and LF* respectively, and 0.3-5.8%, 0.2-3.8%, and 0.1-2.7% for Java considering mid-range smartphones only (G4+ and G5+). Despite of the high bandwidth demands of group video-call, our scheme implementation is able to encrypt in real-time effortlessly.

Our final experiment analyzes the encrypted data volume defined as the percentage of encrypted bits in the compressed video sequence. Our scheme is scalable, so in order to avoid security holes we endorse an optimal range of encryption volume between 0.7-3.0% ($32 \leq BF \leq 128$), with provided security range shown in Table 8. Table 9, compares different selective encryption algorithms in the literature regarding *encrypted information*, *encryption bitrate increase, codec independency, tunable encryption, encrypted data volume, and Complexity Overhead* (ratio between encryption_time/encoding_time). Our scheme represents the minimum scalable data volume securely encrypted and best complexity overhead than any other scheme. Among all analyzed works, Homidouche *et al.* [54] reports the highest speeds for real-time applications in our comparative, delivering 824Mbs on a Core-i5-4300M CPU @ 2.6 GHz. For comparative purposes, we run our proposed cipher on a similar CPU configuration (see Table 10), getting the following speeds: [4000, 2035,1180] Mbs for [HF, MF, LF] encryption modes respectively, representing 30%-80% faster than [54]. It is fair to mention though, that our scheme cannot partially reproduce (decode) entropy coded data without perfect deciphering (this is because headers are subject for encryption as well). However, our scheme has additional important qualities to take care of in data security, such as: a) non-intrusive, b) codec independent (this is why we can easily report encryption performance on different encoders, such as H.264, JPEG2000 and more), and c) do not affect image/video compression ratio (no data overhead is added).

IV. CONCLUSION

We have proposed a new highly effective chaos-based encryption scheme that can handle real-time video communications on a wide gamma of CPU-power capabilities, including low to mid-power smartphone technologies. The aim is to diffuse bit errors along entropy coded bitstreams, so the decoding process is probabilistically speaking not possible. The scheme is entirely based on integer arithmetic that eliminates the Wobbling effect (see section 1) and speed up encryption computation without compromising security. It provides the following advantages:

a) Excellent chaotic properties (based on the NIST test suite); b) Codec independency; it works entirely after entropy coding, therefore input video (or audio, image, etc.), can be in any format (MPEG-4, H.264, etc.); c) Scalable security; d) Fast performance (on C and Java programming languages); and e) Very low encrypting data volume. The scheme was implemented in C and Java programming languages showing the highest performance reported in the literature for smartphones, capable to handle one-to-one to group video calls effortlessly.

REFERENCES

[1] *Number of Smartphone Users From 2016 to 2021*. Accessed: Aug. 30, 2021. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
[2] *Global Mobile Phone Website Traffic Share From 2011 to 2021*. Accessed:Aug. 20, 2020. [Online]. Available: https://www.oberlo.com/statistics/mobile-internet-traffic
[3] C. Marshall. *By 2019, 80% of the World's Internet Traffic Will be Video [Cisco Study]*. Accessed: Mar. 15, 2021. [Online]. Available: https://tubularinsights.com/2019-internet-video-traffic
[4] C.-P. Wu and C.-C. J. Kuo, "Design of integrated multimedia compression and encryption systems," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 828–839, Oct. 2005, doi: 10.1109/TMM.2005.854469.
[5] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 905–917, Oct. 2006, doi: 10.1109/TMM.2006.879919.
[6] D. Xie and C.-C. Kuo, "Multimedia encryption with joint randomized entropy coding and rotation in partitioned bitstream," *EURASIP J. Inf. Secur.*, vol. 2007, no. 1, 2007, Art. no. 035262, doi: 10.1186/1687-417x-2007-035262.

[7] A. Servetti and J. C. D. Martin, "Perception-based partial encryption of compressed speech," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 8, pp. 637–643, Nov. 2002, doi: 10.1109/TSA.2002.804300.

[8] Z. Shahid, M. Chaumont, and W. Puech, "Fast protection of H.264/AVC by selective encryption of CAVLC and CABAC for I and P frames," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 565–576, May 2011, doi: 10.1109/TCSVT.2011.2129090.

[9] A. Said, "Measuring the strength of partial encryption schemes," in *Proc. IEEE Int. Conf. Image Process.*, Genova, Italy, Sep. 2005, p. 1126, doi: 10.1109/ICIP.2005.1530258.

[10] A. Massoudi, F. Lefèbvre, C. D. Vleeschouwer, and F. Devaux, "Secure and low cost selective encryption for JPEG2000," in *Proc. 10th IEEE Int. Symp. Multimedia (ISM)*, Berkeley, CA, USA, Dec. 2008, pp. 31–38, doi: 10.1109/ISM.2008.29.

[11] T. Xiang, K.-W. Wong, and X. Liao, "Selective image encryption using a spatiotemporal chaotic system," *Chaos: Interdiscipl. J. Nonlinear Sci.*, vol. 17, no. 2, Jun. 2007, Art. no. 023115, doi: 10.1063/1.2728112.

[12] S. Som and S. Sen, "A non-adaptive partial encryption of grayscale images based on chaos," *Proc. Technol.*, vol. 10, pp. 663–671, Jan. 2013, doi: 10.1016/j.protcy.2013.12.408.

[13] G. Bhatnagar and Q. M. J. Wu, "Selective image encryption based on pixels of interest and singular value decomposition," *Digit. Signal Process.*, vol. 22, no. 4, pp. 648–663, 2012, doi: 10.1016/j.dsp.2012.02.005.

[14] J. M. Rodrigues, W. Puech, P. Meuel, J. C. Bajard, and M. Chaumont, "Face protection by fast selective encryption in a video," in *Proc. IET Conf. Crime Secur.*, London, U.K., 2006, pp. 420–425.

[15] H. Y. Yang, K. H. Lee, and S. H. Lee, "Method and apparatus for partially encrypting speech packets," U.S. Patent 0 041 231 A1, Feb. 12, 2009.

[16] J. D. Gibson, A. Servetti, H. Dong, A. Gersho, T. Lookabaugh, and J. C. De Martin, "Selective encryption and scalable speech coding for voice communications over multi-hop wireless links," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct./Nov. 2004.

[17] F. Almasalha, R. Hasimoto-Beltran, and A. Khokhar, "Partial encryption of entropy-coded video compression using coupled chaotic maps," *Entropy*, vol. 16, no. 10, pp. 5575–5600, Oct. 2014, doi: 10.3390/e16105575.

[18] F. Dufaux and T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1168–1174, Aug. 2008, doi: 10.1109/TCSVT.2008.928225.

[19] G. Liu, "A selective video encryption scheme for MPEG compression standard," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 89, no. 1, pp. 194–202, Jan. 2006, doi: 10.1093/ietfec/e89-a.1.194.

[20] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm," in *Proc. 6th ACM Int. Conf. Multimedia*, 1998, pp. 81–88.

[21] S. Lian, Z. Liu, Z. Ren, and Z. Wang, "Selective video encryption based on advanced video coding," in *Advances in Multimedia Information Processing-(PCM)* (Lecture Notes in Computer Science), vol. 3768, Y. S. Ho and H. J. Kim, Eds. Berlin, Germany: Springer, 2005, pp. 281–290.

[22] O.-Y. Lui and K.-W. Wong, "Chaos-based selective encryption for H.264/AVC," *J. Syst. Softw.*, vol. 86, no. 12, pp. 3183–3192, Dec. 2013, doi: 10.1016/j.jss.2013.07.054.

[23] L. M. Varlakshmi, G. F. Sudha, and G. Jaikishan, "An efficient scalable video encryption scheme for real time applications," *Proc. Eng.*, vol. 30, pp. 852–860, Jan. 2012, doi: 10.1016/j.proeng.2012.01.937.

[24] S. Yan and Q. Lin, "Partial encryption of JPEG2000 images based on EBCOT," in *Proc. Int. Conf. Intell. Control Inf. Process.*, Aug. 2010, pp. 472–476.

[25] S.-W. Park and S.-U. Shin, "Efficient selective encryption scheme for the H.264/scalable video coding(SVC)," in *Proc. 4th Int. Conf. Netw. Comput. Adv. Inf. Manage.*, Sep. 2008, pp. 371–376.

[26] S. Lian and X. Chen, "On the design of partial encryption scheme for multimedia content," *Math. Comput. Model.*, vol. 57, no. 11, pp. 2613–2624, 2013, doi: 10.1016/j.mcm.2011.06.007.

[27] H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2439–2451, Aug. 2000, doi: 10.1109/78.852023.

[28] P. W. Moo and X. Wu, "Resynchronization properties of arithmetic coding," in *Proc. Int. Conf. Image Process.*, Mar. 1999, p. 540.

[29] G. Zhou and Z. Zhang, "Synchronization recovery of variable-length codes," *IEEE Trans. Inf. Theory*, vol. 48, no. 1, pp. 219–227, Jan. 2002, doi: 10.1109/18.971750.

[30] G. A. Lopez, M. Taufer, and P. J. Teller, "Evaluation of IEEE 754 floating-point arithmetic compliance across a wide range of heterogeneous computers," in *Proc. Conf. Diversity Comput. (TAPIA)*, New York, NY, USA, 2007, pp. 1–4.

[31] J. Maxted and J. Robinson, "Error recovery for variable length codes," *IEEE Trans. Inf. Theory*, vol. 31, no. 6, pp. 794–801, Nov. 1985, doi: 10.1109/TIT.1985.1057110.

[32] B. L. Montgomery and J. Abrahams, "Synchronization of binary source codes," *IEEE Trans. Inf. Theory*, vol. 32, no. 6, pp. 849–854, Nov. 1986, doi: 10.1109/TIT.1986.1057238.

[33] M. Monaco and J. Lawler, "Corrections and additions to 'error recovery for variable length codes' by J.C. maxted and J.P. robinson," *IEEE Trans. Inf. Theory*, vol. 33, no. 3, pp. 454–456, May 1987, doi: 10.1109/TIT.1987.1057297.

[34] P. F. Swaszek and P. DiCicco, "More on the error recovery for variable-length codes," *IEEE Trans. Inf. Theory*, vol. 41, no. 6, pp. 2064–2071, Nov. 1995, doi: 10.1109/18.476338.

[35] Y. Takishima, M. Wada, and H. Murakami, "Error states and synchronization recovery for variable length codes," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 783–792, Feb. 1994, doi: 10.1109/TCOMM.1994.577107.

[36] P. M. Binder and R. V. Jensen, "Simulating chaotic behavior with finite-state machines," *Phys. Rev. A, Gen. Phys.*, vol. 34, no. 5, pp. 4460–4463, 1986.

[37] W. Wang, Z. Liu, and B. Hu, "Phase order in chaotic maps and in coupled map lattices," *Phys. Rev. Lett.*, vol. 84, no. 12, p. 2610, 2000.

[38] Y. Dobyns and H. Atmanspacher, "Characterizing spontaneous irregular behavior in coupled map lattices," *Chaos, Solitons Fractals*, vol. 24, no. 1, pp. 313–327, Apr. 2005.

[39] H. Hu, Y. Xu, and Z. Zhu, "A method of improving the properties of digital chaotic system," *Chaos, Solitons Fractals*, vol. 38, no. 2, pp. 439–446, 2008.

[40] F. Xie and G. Hu, "Clustering dynamics in globally coupled map lattices," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 56, pp. 1567–1576, Aug. 1997.

[41] R. Matthews, "On the derivation of a chaotic encryption algorithm," *Cryptologia*, vol. 13, pp. 29–42, Jan. 1989.

[42] M. S. Baptista, "Cryptography with chaos," *Phys. Lett. A*, vol. 240, pp. 50–54, Mar. 1998.

[43] S. Li, X. Mou, and Y. Cai, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography," in *Progress in Cryptology–(INDOCRYPT)* (Lecture Notes in Computer Science), vol. 2247. Berlin, Germany: Springer, 2001, pp. 316–329.

[44] R. Hasimoto-Beltran, "Freac-LUT: A new dynamic look-up table approach to secure chaotic encryption," *Int. J. Bifurcation Chaos*, vol. 23, no. 1, 2013, Art. no. 1350004.

[45] Z. Neufeld and T. Vicsek, "Spatiotemporal chaos in a coupled map lattice with unstable couplings," *J. Phys. A, Math. Gen.*, vol. 28, no. 18, pp. 5257–5266, Sep. 1995.

[46] X. Wang, N. Zheng, and L. Tian, "Hash key-based video encryption scheme for H.264/AVC," *Signal Process., Image Commun.*, vol. 25, no. 6, pp. 427–437, Jul. 2010.

[47] Z. Fawaz, H. Noura, and A. Mostefaoui, "Lightweight format-compliant encryption algorithm for JPEG 2000 images," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1871–1878.

[48] T. Addabbo, M. Alioto, A. Fort, A. Pasini, S. Rochhi, and V. Vignoli, "A class of maximum-period nonlinear congruential generators derived from the Rényi chaotic map," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 4, pp. 816–828, Apr. 2007.

[49] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST, Gaithersburg, MD, USA, Tech. Rep. SP 800-22, 2010.

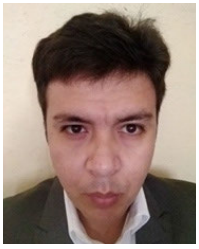
[50] Microsoft. *How Much Bandwidth Does Skype Need*. Accessed: Jan. 2018. [Online]. Available: https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need

[51] *Low- to Mid-Range Smartphones Dominate Worldwide Smartphone*. Accessed: Sep. 25, 2020. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS46865120

[52] N. Dolati, A. Beheshti, and H. Azadegan, "A selective encryption for H.264/AVC videos based on scrambling," *Multimedia Tools Appl.*, vol. 80, pp. 2319–2338, Sep. 2021, doi: 10.1007/s11042-020-09654-3.

[53] X. Di, Y. Wang, J. Li, L. Cong, H. Qi, and Y. Zhang, "An optimized video selective encryption algorithm," in *Proc. 10th Int. Congr. Image Signal Process., Biomed. Eng. Informat. (CISP-BMEI)*, Oct. 2017, pp. 1–5.
[54] W. Hamidouche, M. Farajallah, N. Sidaty, S. El Assad, and O. Deforges, "Real-time selective video encryption based on the chaos system in scalable HEVC extension," *Signal Process., Image Commun.*, vol. 58, pp. 73–86, Oct. 2017, doi: 10.1016/j.image.2017.06.007.
[55] Y. Chung, S. Lee, T. Jeon, and D. Park, "Fast video encryption using the H.264 error propagation property for smart mobile devices," *Sensors*, vol. 15, no. 4, pp. 7953–7968, Apr. 2015, doi: 10.3390/s150407953.
[56] S. Cheng, L. Wang, N. Ao, and Q. Han, "A selective video encryption scheme based on coding characteristics," *Symmetry*, vol. 12, no. 3, p. 332, Feb. 2020, doi: 10.3390/sym12030332.
[57] R. A. Shah, M. N. Asghar, S. Abdullah, N. Kanwal, and M. Fleury, "SLEPX: An efficient lightweight cipher for visual protection of scalable HEVC extension," *IEEE Access*, vol. 8, pp. 187784–187807, 2020, doi: 10.1109/ACCESS.2020.3030608.
[58] I. Hraini, M. Farajallah, N. Arman, and W. Hamidouche, "Joint crypto-compression based on selective encryption for WMSNs," *IEEE Access*, vol. 9, pp. 161269–161282, 2021, doi: 10.1109/ACCESS.2021.3131566.

ROGELIO HASIMOTO-BELTRAN (Member, IEEE) received the Ph.D. degree in computer and electrical engineering from the University of Delaware, USA, in 2001. After his Ph.D., he spent two years at Akamai Technologies (a leader enterprise in multimedia content delivery), as a Senior Software Engineer. In 2003, he joined the Department of Computer Science, Center for Research in Mathematics (CIMAT), Mexico, where he was a tenure and promoted to a Senior Research Scientist. He was a Visiting Associate Professor with the University of Illinois at Chicago, from 2009 to 2010. He has published more than 45 technical papers in refereed conferences and journals in the area of image processing, computer vision, and multimedia networks. His current research interests include robust multimedia communication, error concealment, face detection and recognition, chaotic encryption, and machine learning applications.



MARCOS D. CALDERON-CALDERON received the B.S. degree in computer systems from the National Technological Institute of México, in 2012, and the M.Sc. degree in computer science from the Center for Research in Mathematics (CIMAT), Guanajuato, Mexico, in 2014. From 2015 to 2021, he worked as a Software Engineer at Teknei S.A. de C.V., Mexico City. His research interests include machine learning, human–computer interaction, and fingerprint recognition.

VÍCTOR H. OLAVARRÍA-JARAMILLO received the B.S. degree in mechatronic and computer vision engineering from the "Instituto Superior de Irapuato," Guanajuato, Mexico, in 2015. Since 2017, he has been with the Center for Research in Mathematics (CIMAT) as a Senior Software Engineer, where he develops modular Intelligent software for public transportation in embedded system and android monitoring applications. His current research interest includes machine learning applications to environmental hazards.