This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2017.2715659, IEEE Transactions on Circuits and Systems II: Express Briefs

Secure and Lightweight Compressive Sensing using Stream Cipher

Vikramkumar Pudi, Anupam Chattopadhyay *Senior Member, IEEE*, and Kwok-Yan Lam *Member, IEEE*

***Abstract*—We are proposing a lightweight and secure compressive sensing method for compression of data generated by image sensors. This method simultaneously compresses and encrypts image and video files, thereby reducing communication cost, storage space, runtime and power/energy overhead. Our proposed method combines the compressive sensing technique with stream cipher to implement the secure compressive sensing. Furthermore, the usage of stream cipher to generate the measurement matrix results in minimal overhead to achieve the security objective. Our experimental studies using FPGA prototype show that the security enhancements to the circuit comes with no timing overhead, reduces 41%-84% of memory requirements and only 8%-33% increase in the LUTs depending on the compression ratio.**

***Index Terms*—Compressive Sensing, Stream Cipher, Security**

I. Introduction

Continuous and rapid strides in the innovation of the electronics and telecommunication industry has made pervasive computing a ground reality - leading to sensor-infested smart infrastructures, such as, Internet of Things (IoT). With growing number of smart sensors and connected devices, communication bandwidth and storage is becoming a scarce resource. Moreover, communication of sensitive and private information over public communication channels raise the possibility of significant economic and infrastructural risks. Therefore, security and compression of sensory data, in particular image/video, is a challenging concern.

Compressive Sensing (CS) is an emerging technology, which has attracted lots of interest in signal processing and wireless communication networks during the last decade. Essentially, it is a method of compressing signals using less resources [1], [2], [3], [4] than required by a perfect lossless technique. However, reconstruction of compressed signal is a computationally demanding task, due to which, CS reconstruction is usually undertaken at servers/base stations. CS uses a Φ-matrix, also called measurement matrix, for compression of the original signal. If this Φ-matrix is a secret known only to the sender and recipient, it can be used as as the *key* for encrypting and compressing the images or signals simultaneously. However, compression and encryption using Φ-matrix does not guarantee security, which can be easily validated by studying the information leakage and image histograms.

Secure encryption of signal using CS have been studied and presented in [5], [6], [7], [8], [9]. Security of the transmitted signal is especially relevant for scenarios like IoT, personalized healthcare and robotics. It is to be noted that a prime use case of CS is compression of electrocardiography (EKG) [4] or electroencephalogram (EEG) signals [3], for which confidentiality is a major concern. In [10] and [11], the authors proposed Linear Feedback Shift Register (LFSR) based methods for Secure Compressive Sensing (SCS). The LFSR is used to generate the Φ-matrix and that itself is treated as a secret-key. However, these techniques falls short of any security guarantee, which is also demonstrated by our study of image histograms in section-III.

In this work, we propose a new method for secure encryption using the combination of stream cipher and compressive sensing. First, we study the properties of Bernoulli random matrix to be used as the measurement matrix in the CS. Second, we show that an LFSR-based stream cipher can be used to generate such a matrix. Third, we reuse the stream cipher for performing encryption. Fourth, we schedule the operations in such a way to ensure minimal area overhead and completely avoid the storage requirements for the Φ-matrix. Finally, we validate our technique with empirical studies and present implementation results.

Furthermore, our method is designed with flexibility to allow trade-off between security and computing overhead by only performing partial encryption on the compressed samples.

Related Work

The CS compression and encryption of images presented in [6] uses the logistic chaos map based key control circulant matrix as measurement matrix. The measurement matrix generation in [6] involves floating point number multiplications, additions, shifts, swap and comparison operations. This method requires more computational resources and not suitable for resource constrained devices.

The CS encryption in [12] provides two layer security by randomly permuting and generating the measurement matrix, which is also difficult to satisfy the limited resource constraint.

LFSR based measurement-matrix generation are presented in [10], [11] and [13]. The LFSR based measurement matrix is used for compression of signals using CS. Even though the methods in [10], [11] and [13] are lightweight, these fall short of security guarantee. Our studies shows that after CS compression, such schemes still need to use additional encryption layer for attaining confidentiality.

II. Compressive Sensing and Stream Ciphers

A. Block-based Compressive Sensing

Compressive sensing is a method of directly acquiring the signal at sub-Nyquist rate by sparse sampling from the sensors, thus creating a $sparse$-signal. Compressive sensing acquires M-samples signal (Y) from an N-samples signal (X) using the measurement matrix Φ of size $M \times N$ provided $M << N$.

Vikramkumar Pudi, Anupam Chattopadhyay, and Kwok-Yan Lam are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore; e-mail: pudi@ntu.edu.sg.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2017.2715659, IEEE Transactions on Circuits and Systems II: Express Briefs

This process is called CS compression or CS encoding. The CS encoding is shown in Eq. 1, provided X is a sparse signal.

$$Y_{M\times 1} = \Phi_{M\times N} X_{N\times 1} \qquad \text{Where M} \ll \text{N} \qquad (1)$$

We can apply Basis-Pursuit (BP) or l_1-optimization method for recovering signal. l_1-optimization is defined as the following.

$$X = arg\ min \parallel X \parallel_1 \text{ subjected to } Y = \Phi X \qquad (2)$$

In general sensor signals are not sparse. However, one can represent such signals as sparse by using particular basis like Discrete Wavelet, Discrete Cosine etc. known as Ψ. If signal X is sparse in some basis Ψ, then we can directly compute the Y using the Eq. 1 without transforming the X. The signal recovery method in Eq. 2 changed to Eq. 3

$$X = arg\ min \parallel X \parallel_1 \text{ subjected to } Y = \Phi\Psi X \qquad (3)$$

where X, Φ, Ψ and Y are $N \times 1$, $M \times N$, $N \times N$ and $M \times 1$ matrices or signals respectively.

The 2-D signals like images must be converted to 1-D signals before applying the CS compression in Eq. 1. Recovering 2-D signal using the BP in Eq. 3 is computationally expensive because of large value of N. The image is split into several blocks of size $B \times B$ and CS is applied on each block [14]. Further, each block is recovered by using the method in Eq. 3. This method is called as Block-based Compressive Sensing (BCS),which is ideal for lightweight construction. Further, we adopted the Smoothed Projected Landweber (SPL) method for reconstructing BCS based compressed images [15]. The BCS technique also fits well with the idea of stream cipher-based measurement matrix generation.

B. Stream Ciphers

Stream ciphers are symmetric-key ciphers. It generates pseudo-random sequence known as $Key\text{-}Stream$ using a secret key and then produces $Cipher\text{-}text$ by XOR-ing individual bit of $Key\text{-}Stream$ with message bit. Similarly stream cipher re-generates the original message from $Cipher - text$ using the same $Key - Stream$. The block diagram of stream cipher is shown in Fig. 1. Due to lower resource requirement and faster operations, stream ciphers are preferred as lightweight cryptographic primitive over block ciphers.

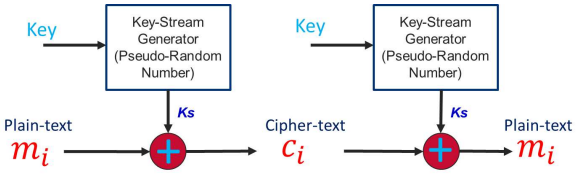


Fig. 1: Stream Cipher Encryption and Decryption

C. Measurement Matrix Generation using Stream Cipher

The robustness of CS strongly depends on the selection of a good measurement matrix Φ for exact recovery of X from Y. In [16], the authors introduced the restricted isometry property (RIP) as a sufficient condition on Φ-matrices for recovering signal from compressed signal. Gaussian random matrices, Partial Fourier matrices and Bernoulli matrices are possible matrices that satisfy the RIP condition of good measurement matrices.

The Bernoulli random matrices [12] contains $+1$ and -1 values with probabilities of 0.5. We can write Bernoulli random matrices as following.

$$\phi_{ij} = \begin{cases} +\frac{1}{\sqrt{N}} & \text{with probability 1/2} \\ -\frac{1}{\sqrt{N}} & \text{with probability 1/2} \end{cases}$$

Bernoulli random matrices require only adders for implementing CS encoder in Eq. 1, thereby making it an attractive choice for lightweight CS implementation.

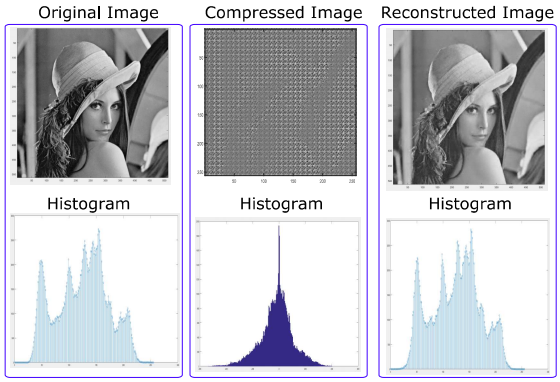


Fig. 2: Original, compressed (75%) and reconstructed images with histograms. Φ-matrix generated using stream cipher.

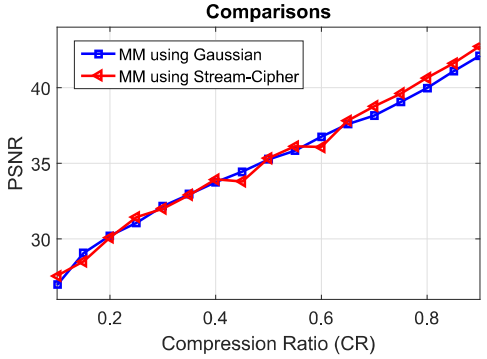


Fig. 3: Comparisons of PSNRs of different Φ-matrices generated using Gaussian and Stream Cipher

Stream cipher generates the $key\text{-}Stream$ bits 0 and 1 with a probability of 0.5, and therefore, naturally qualifies for Bernoulli matrix. We use $keystream$ bits as the Bernoulli random matrices, where 0 and 1 represents $+1$ and -1 respectively. There is a wide choice of lightweight stream ciphers, such as, Grain [17], and Trivium [18], from which, we have deployed Trivium for our implementation. The Fig. 2 shows the compressed and reconstructed images using Φ-matrix generated Trivium. Fig. 3 shows comparisons of probabilities of signal to noise ratios (PSNR) of reconstructed and original images using different Φ-matrices. The PSNRs of reconstructed images using stream cipher-based Φ-matrices

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2017.2715659, IEEE Transactions on Circuits and Systems II: Express Briefs

are almost same as using Gaussian Φ-matrices.Due to our adoption of block-based compressive sensing technique, the number of samples compressed can be tuned with the rate of generating matrix elements by the stream cipher. This completely rules out the storage requirement for the Φ-matrix making the overall design very compact.

III. Secure Compressive Sensing Architecture

In this section, we explore two alternative methods for Secure Compressive Sensing (SCS). First one is implementation of the CS encoder using the stream cipher-based Φ-matrix, which is referred as CS using Stream Cipher (CSSC). Second one is encrypting the Most Significant bits of each sample generated from CSSC stage using the same stream cipher.

A. CS encoder using the Stream Cipher

The CS encoder in Eq. 1 compresses the N-samples to M-samples. The $N = B^2$ in case of BCS, where size of block is $B \times B$. We are defining the compression ration (CR) as

$$CR = (1 - \frac{M}{N})100\%$$

The elements of Φ-matrix are -1s and $+1$s, which allows the implementation of the CS encoder using only adders and subtractors. The i^{th} column of Φ-matrix is shown in Eq. (4). The $\phi_1(i)$ represents the first row's i^{th} element of Φ-matrix.

$$\Phi(i) = [\phi_1(i)\ \phi_2(i) \dots \phi_M(i)]^T \quad (4)$$

We use a stream cipher to generate one column of the Φ-matrix at each clock cycle. Assuming the recording of each sample at a time, it is possible to add one sample to the corresponding element in Φ-matrix at a time. The Fig. 4 shows the proposed architecture for CS encoder using the stream cipher based Φ-matrix. The proposed architecture requires only M number of adder/subtractor based accumulators in addition to the stream cipher.

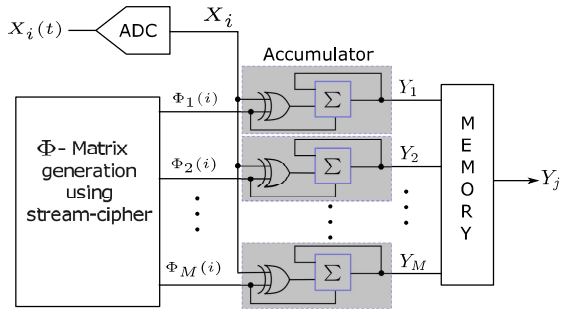


Fig. 4: Proposed Architecture of Compressive Sensing with Stream Cipher-based Φ-matrix

The Eq. (5) shows the output of accumulator Y_1 at i^{th} iteration. The X_i is subtracted from the previous accumulator value Y_1^{i-1} if $\phi_1(i) = 1$ otherwise X_i is added to the previous value. The accumulator can be implemented using only adder with input carry and XOR gates as shown in Eq. (6).

$$Y_1^i = Y_1^{i-1} + (-1)^{\phi_1(i)} X_i \quad (5)$$

$$= Y_1^{i-1} + \phi_1(i) \oplus X_i + \phi_1(i) \quad (6)$$

Initially the stream cipher generates the M elements or one-column of Φ-matrix. The accumulator takes each input sample X_i at a time and adds or subtracts from the previous accumulator value depending upon the Φ-matrix elements. This process continues until the completion of image block, which takes $N(= B^2)$ number of clock cycles.

Stream cipher requires a secret key, without which one cannot generate the same random sequence or Φ-matrix. This prevents reconstruction of original signal from compressed signal without the correct secret-key. The Fig. 5 shows the reconstructed images using the different Φ-matrix than original Φ-matrix. We refer to this method as compressive sensing with stream cipher (CSSC).

Fig. 5: Reconstructed images from compressed images using different Φ-matrix (block-size = 16×16 and CR= 75%)

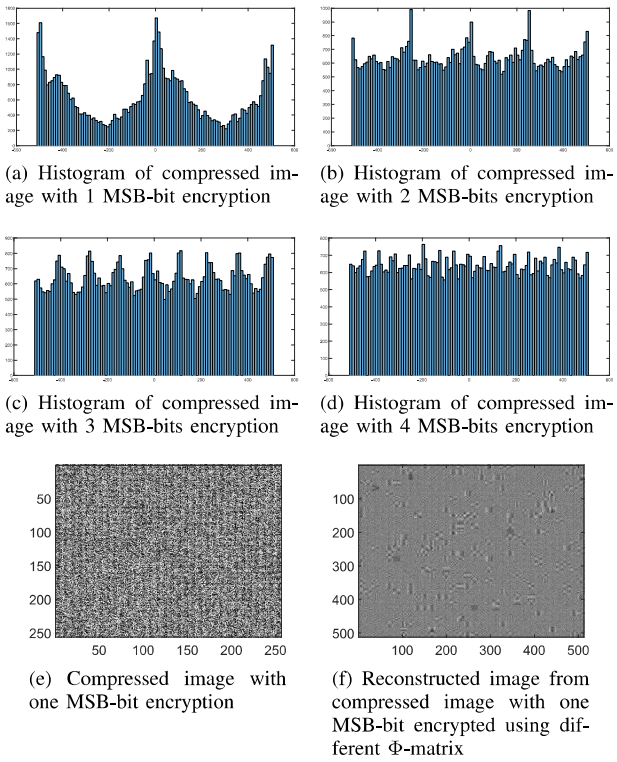
From Fig. 5, we can observe that the reconstructed images are not same as the original ones, which is due to the incorrect Φ-matrix. Still the reconstructed images in Fig. 5 leaks information about the original images, since the CS encoder preserves some of the original signal properties. This is the same issue experienced by previously proposed secure compressive sensing methods in [6], [11] and [10], which were based on LFSR. In the next section, we propose an alternative method for CSSC without increasing the hardware complexity.

B. CSSC stage with encryption of MSB-bits of Compressed samples

Compressed signals generated from CS preserves some of the original signal properties due to the linear combination of input samples as shown in Eq. (1) and generation of non-uniform histogram as shown in Fig. 2. Hence, one needs to encrypt the compressed samples generated from CSSC stage for preventing leakage of information. Instead of a full encryption of the compressed sample, we observe that a partial encryption suffices to prevent information leakage. This is due to the fact that, the MSB-bits of compressed samples Y_js contains more useful information than the LSBs. Our empirical studies show that encryption of 4 MSBs (most significant half-byte/nibble) of compressed samples Y_js provides an uniform histogram and does not leak any information about original

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2017.2715659, IEEE Transactions on Circuits and Systems II: Express Briefs

image while transmitting or reconstructing using the different Φ-matrix as shown in Fig. 6.



(a) Histogram of compressed image with 1 MSB-bit encryption

(b) Histogram of compressed image with 2 MSB-bits encryption

(c) Histogram of compressed image with 3 MSB-bits encryption

(d) Histogram of compressed image with 4 MSB-bits encryption

(e) Compressed image with one MSB-bit encryption

(f) Reconstructed image from compressed image with one MSB-bit encrypted using different Φ-matrix

Fig. 6: Histograms of compressed and reconstructed image using different Φ-matrices

The Fig. 6 shows the histograms of compressed image with various number of MSBs encryption. From Fig. 6(d), we can observe that encrypting 4 MSB-bits of compressed samples provides the uniform histogram, which is essential requirement for the encryption. The compressed image with encryption and the reconstructed image using a different Φ-matrix are shown in Fig. 6(e) and 6(f) respectively. This method is termed as CSSC with encrypting MSBs (CSSC-EM).

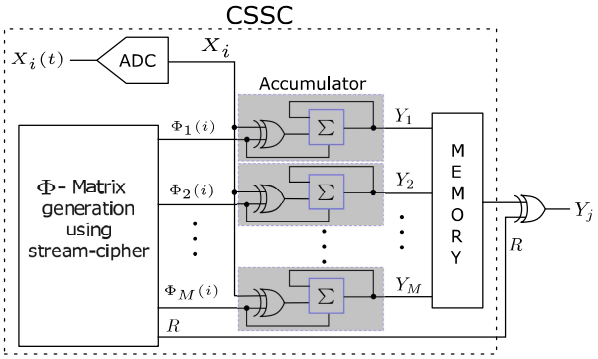


Fig. 7: Proposed Architecture of Compressive Sensing using Stream Cipher-based Φ-matrix with MSB-bits encryption of compressed samples Y_js

The proposed architecture for the CSSC-EM is shown in Fig. 7. The dotted rectangle box in Fig. 7 represents the CSSC stage. The MSBs of compressed samples are encrypted using the pseudo random sequence R generated from stream cipher. Note that, R is generated by the same stream cipher used for generating the Φ-matrix, and we take the character size of R as 4-bit, which requires only additional four xor gates for encrypting the compressed data compared to the CSSC method. This method with partial encryption allows users to perform dynamic trade-off between security and resource requirements. The following steps are involved in the operation of proposed CSSC-EM method.

- *Step I:* Initialization of stream cipher using the secret-key and reading image pixel by pixel.
- *Step II:* Generating M random bits of Φ-matrix and four-bits of R using the stream cipher.
- *Step III:* Multiplying image pixel to the corresponding elements in Φ-matrix and adding to the previous accumulator value for generating the compressed samples as Eq. 6.
- *Step IV:* After completion of compression, all the compressed samples are stored as single image block in memory.
- *Step V:* Reading each sample from the memory at a time and encrypting each compressed sample using the pseudo-random sequence R.

IV. Experimental Studies

The BCS method is applied block-wise on the incoming input image. Considering an image of size $S1 \times S2$ split into L-number of blocks, where size of each block is $B \times B$. Each pixel in the image is represented with one-byte (gray-scale format) and each block compression processes executes B^2 bytes. The B^2 number of pixels are multiplied with Φ matrix for generating the compressed samples of size M. The size of Φ matrix is $M \times B^2$, where $M \ll B^2$.

The BCS method requires M-number of accumulators for generating the M compressed samples as shown in Fig. 4 and 7. We are representing the size and critical path of each accumulator as W-bits and T_a. The latency for generating the M number of compressed samples are B^2 clock-cycles. The memory required for storing the M compressed samples are MW-bits. In BCS we have to store the Φ-matrix, and therefore, the overall memory requirement is sum of MB^2-bits and MW-bits. After compression of one block, BCS will start compressing another block by reading each pixel at a time and simultaneously it will transmit one compressed sample (of size W-bits) at a time by reading from the memory. All the elements in memory are transmitted before completion of other block compression, because $M \ll N$.The total time required for the compression of whole image is LB^2 number of clock-cycles and additional M clock-cycles for transmission of compressed samples. The total compression time (TCT) is $LB^2 + M$ cycles. The maximum critical path of BCS depends on critical path of accumulator (T_a), which can be easily observed from Fig. 4 and 7.

The area and time complexities of the proposed CSSC and CSSC-EM architectures are similar to the BCS, except that these methods use the stream cipher to generate the Φ-matrix and thus, avoids the storage space required for Φ-matrix. The CSSC-EM requires additional 4 registers and 4 XOR gates for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2017.2715659, IEEE Transactions on Circuits and Systems II: Express Briefs

storing R and encrypting MSB-bits of compressed samples Y_j. In the worst case (full encryption of compressed samples) CSSC-EM requires additional W registers and W Xor gates (where W is bit size of Y_j).

TABLE I: Area-Delay complexity of BCS, CSSC, CSSC-EM

Algorithm	Accumulators (W-bits)	Memory (bits)	Latency (Cycles)	Through-put
BCS	M	$MW + MB^2$	B^2	W-bits
CSSC	M	MW	B^2	W-bits
CSSC-EM	M	$MW + 4$	B^2	W-bits

TABLE II: Trivium Hardware Complexity [18]

Components	1-bit	8-bit	16-bit	32-bit	64-bit
Memory(bits)	288	288	288	288	288
XOR gates	11	88	176	352	704
AND gates	3	24	48	96	192

The Table I shows theoretical area and delay complexity comparisons of BCS, CSSC and CSSC-EM without the stream cipher. For the sake of comparison, we have also provided stream cipher (Trivium) hardware complexities in Table II. The delay complexities of CSSC-EM is same as the BCS. The CSSC-EM requires more logic gates (XOR and AND) than BCS but it requires less memory (MB^2-bits) than BCS. For example block-size 8×8 and M = 16, CSSC-EM requires 1020 memory bits less than the BCS but it requires additional 448 gates (Trivium with 32-bit output) compared to the BCS. The area requirement for the memory is more than logic gates, because of which proposed CSSC-EM requires less area than the BCS.

Table III shows the FPGA synthesis results of BCS, CSSC and CSSC-EM architecture on Artix-7 board with block-size 8×8 for different compression ratios. Note that the area overhead can be different depending on the choice of the stream cipher. The BCS method requires additional RAM for storing the Φ-matrix. The proposed CSSC-EM architecture saves 41%-84% of memory (including registers and RAM) compared to BCS, while it requires only 8%-33% more LUTs for implementing compression and encryption simultaneously.

TABLE III: Synthesis Results

Design	Logic Utilization	Compression Ratio (%)				
		90	80	70	60	50
	Block-Size 8×8					
BCS	LUTs	162	271	393	502	690
	Registers	238	395	578	734	610
	RAM(bits)	448	832	1280	1664	2048
	Speed (MHz)	371.5	366	352.6	352.6	352.2
CSSC	LUTs	251	360	519	634	748
	Registers	404	662	853	1003	1154
	Speed (MHz)	371.5	366	352.6	352.6	352.2
CSSC-EM	LUTs	255	364	523	650	752
	Registers	404	662	853	1003	1154
	Speed (MHz)	371.5	366	331.6	331.6	319.3

V. Conclusion

In this paper, the idea of compressive sensing coupled with encryption for image and video is proposed. This technique comes with low overhead due to reuse of the encryption module and the generation of measurement matrix. We show that standalone LFSR-based measurement matrix generation does not provide any confidentiality guarantee and hence, additional encryption layer is necessary.

The applications of our proposed method includes, among others, real-time monitoring systems, secure transmission of medical data (like ECG signals, MR images), traffic videos, environmental monitoring, Unmanned Areal Vehicles (UAVs) and IoT sensors, for which, in future, we will like to make case specific studies on the accuracy of video/image data analytics.

References

[1] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, pp. 21–30, 2008.

[2] X. Chen, Z. Yu, S. Hoyos, B. M. Sadler, and J. Silva-Martinez, "A sub-nyquist rate sampling receiver exploiting compressive sensing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 3, pp. 507–520, March 2011.

[3] M. Shoaib, K. H. Lee, N. K. Jha, and N. Verma, "A 0.6107μw energy-scalable processor for directly analyzing compressively-sensed eeg," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1105–1118, April 2014.

[4] F. Chen, F. Lim, O. Abari, A. Chandrakasan, and V. Stojanovic, "Energy-aware design of compressed sensing systems for wireless sensors under performance and reliability constraints," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 3, pp. 650–661, March 2013.

[5] Y. Zhang, J. Zhou, F. Chen, L. Y. Zhang, K.-W. Wong, X. He, and D. Xiao, "Embedding cryptographic features in compressive sensing," *Neurocomputing*, 2016.

[6] N. Zhou, A. Zhang, F. Zheng, and L. Gong, "Novel image compression–encryption hybrid algorithm based on key-controlled measurement matrix in compressive sensing," *Optics & Laser Technology*, vol. 62, pp. 152–160, 2014.

[7] V. Cambareri, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Low-complexity multiclass encryption by compressed sensing," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2183–2195, 2015.

[8] X. Zhang, Y. Ren, G. Feng, and Z. Qian, "Compressing encrypted image using compressive sensing," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), International Conference on*. IEEE, 2011, pp. 222–225.

[9] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, "Privacy-assured outsourcing of image reconstruction service in cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 166–177, 2013.

[10] R. Dautov and G. R. Tsouri, "Establishing secure measurement matrix for compressed sensing using wireless physical layer security," in *Computing, Networking and Communications (ICNC), International Conference on*. IEEE, 2013, pp. 354–358.

[11] S. N. George and D. P. Pattathil, "A secure LFSR based random measurement matrix for compressive sensing," *Sensing and Imaging*, vol. 15, no. 1, pp. 1–29, 2014.

[12] G. Zhang, S. Jiao, X. Xu, and L. Wang, "Compressed sensing and reconstruction with bernoulli matrices," in *Information and Automation (ICIA), IEEE International Conference on*. IEEE, 2010, pp. 455–460.

[13] F. Chen, A. P. Chandrakasan, and V. Stojanovic, "A signal-agnostic compressed sensing acquisition system for wireless and implantable sensors," in *IEEE Custom Integrated Circuits Conference*, Sept 2010, pp. 1–4.

[14] L. Gan, "Block compressed sensing of natural images," in *15th International conference on digital signal processing*. IEEE, 2007, pp. 403–406.

[15] S. Mun and J. E. Fowler, "Block compressed sensing of images using directional transforms," in *16th IEEE international conference on image processing (ICIP)*. IEEE, 2009, pp. 3021–3024.

[16] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9, pp. 589–592, 2008.

[17] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86–93, 2007.

[18] C. De Canniere and B. Preneel, "Trivium," in *New stream cipher designs*. Springer, 2008, pp. 244–266.