

On Compression of Data Encrypted with Block Ciphers

Demijan Klinc^{*}, Carmit Hazay[†], Ashish Jagmohan, Hugo Krawczyk
and Tal Rabin[‡]

^{*}Georgia Institute of Technology, Atlanta, GA, USA
Email: demi@ece.gatech.edu

[†]Bar-Ilan University, Ramat-Gan, Israel
Email: harelc@cs.biu.ac.il

[‡]IBM T.J. Watson Research Labs, Yorktown Heights and Hawthorne, NY, USA
Email: {ashishja, talr}@us.ibm.com, hugo@ee.technion.ac.il

Abstract

This paper investigates compression of encrypted data. It has been previously shown that data encrypted with Vernam’s scheme [1], also known as the one-time pad, can be compressed without knowledge of the secret key, therefore this result can be applied to stream ciphers used in practice. However, it was not known how to compress data encrypted with non-stream ciphers. In this paper, we address the problem of compressing data encrypted with block ciphers, such as the Advanced Encryption Standard (AES) used in conjunction with one of the commonly employed chaining modes. We show that such data can be feasibly compressed without knowledge of the key. We present performance results for practical code constructions used to compress binary sources.

1 Introduction

We consider the problem of compressing encrypted data. Traditionally in communication systems, data from a source is first compressed and then encrypted before it is transmitted over a channel to the receiver. While in many cases this approach is befitting, there exist scenarios where there is a need to reverse the order in which data encryption and compression are performed. Consider for instance a network of low-cost sensor nodes that transmit sensitive information over the internet to a recipient. The sensor nodes need to encrypt data to hide it from potential eavesdroppers, but they do not necessarily want to compress it as that would require additional hardware and thus higher implementation cost. On the other hand, the network operator that is responsible for transferring the data to the recipient wants to compress the data to maximize the utilization of its resources. It is important to note that the network operator is not trusted and hence does not have access to the key used for encryption

and decryption of data. If it had the key, it could simply decrypt data, compress and encrypt again.

We focus on compression of encrypted data where the encryption procedure utilizes block ciphers such as the Advanced Encryption Standard (AES) [2] and Data Encryption Standard (DES) [3]. Loosely speaking, block ciphers operate on inputs of fixed length and serve as important building blocks that can be used to construct secure encryption schemes.

Block ciphers with a fixed key are a bijection, therefore the entropy of an input is the same as that of the output. It follows that it is theoretically possible to compress the source to the same level as before encryption. However, in practice, encrypted data appears to be random and the conventional compression techniques do not yield desirable results. Consequently, it was long believed that encrypted data is practically incompressible. In a surprising paper [4], the authors break that paradigm and show that the problem of compressing one-time pad encrypted data translates to the problem of compressing correlated sources, which was solved by Slepian and Wolf in [5] and for which practical and efficient codes are known. Compression is practically achievable due to a simple symbol-wise correlation between the key (one-time pad) and the encrypted message. However, when such correlation is more complex, as is the case with block ciphers, the approach to Slepian-Wolf coding utilized in [4] is not directly applicable.

In this paper we investigate if data encrypted with block ciphers can be compressed without access to the key. We show that block ciphers in conjunction with the most commonly used chaining modes in practice [6, 7] are practically compressible for some types of sources. To our knowledge this is the first work to show that non-negligible compression gains can be achieved for cryptographic algorithms like AES or DES when they are used in non-stream modes; in particular, this work offers a solution to the open problem formulated in [8, Sec. 3.3].

The outline of this paper is as follows. Section 2 defines the problem that we seek to solve and summarizes existing work on the subject. Section 3 focuses on block ciphers and explains how they can be compressed without knowledge of the secret key. Subsequently in Section 4 we present the simulation results. The last section gives a brief summary of the presented work.

2 Preliminaries

We begin with a formal definition of an encryption scheme as stated in [9]. A private-key encryption scheme is a triple of algorithms (Gen, E, D) , where Gen is a probabilistic algorithm that outputs a key K chosen according to some distribution that is determined by the scheme; the encryption algorithm E takes as input a key K and a plaintext message X and outputs a ciphertext $E_K(X)$; the decryption algorithm D takes as input a key K and a ciphertext $E_K(X)$ and outputs a plaintext X .

In private-key encryption schemes the same key is used for encryption and decryption algorithms. Private-key encryption schemes can be divided in two categories: block ciphers and stream ciphers. Stream ciphers encrypt plaintext one symbol at

a time, typically by summing it with a key (XOR operation for binary alphabets). In contrast, block ciphers represent a different approach where encryption is accomplished by means of nonlinear mappings on input blocks of fixed length. Common examples of block ciphers are the AES and DES. Typically, block ciphers are not used as a stand-alone encryption procedure but are rather combined to work on variable-length data using composition mechanisms known as chaining modes or modes of operation. The most common mode of operation is cipher-block chaining (CBC) which we describe in Section 3.

We proceed with a formulation of the source coding problem with decoder side-information, which is illustrated in Figure 1. Consider random variables X (termed the source), and Y (termed the side-information), both over a finite-alphabet and with a joint probability distribution P_{XY} . Consider a sequence of independent n realizations of (X, Y) denoted by $\{X_i, Y_i\}_{i=1}^n$.

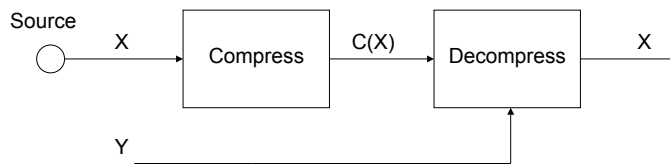


Figure 1: Lossless source coding with decoder side-information.

The problem at hand is of losslessly encoding $\{X_i\}_{i=1}^n$, with $\{Y_i\}_{i=1}^n$ known only to the decoder. In [5], Slepian and Wolf showed that, asymptotically in block-length, this can be done at rates arbitrarily close to the conditional entropy $H(X|Y)$. Practical Slepian-Wolf coding schemes use constructions based on good linear error-correcting codes [10, 11].

Of interest to us, in this paper, are systems which perform both compression and encryption, wherein only the encryptor has access to the key. Typically, in such systems, encryption is performed after compression as depicted in Figure 2(a). This is a consequence of the traditional view which considers ciphertext data hard to compress without knowledge of the key. In [4] a system similar to Figure 2(b) is considered instead, in which the order of the encryption and compression operations at the encoder is reversed¹. The authors consider encryption of a plaintext X using a one-time pad scheme, with a finite-alphabet key (pad) K , to generate the ciphertext $E_K(X)$, i.e.

$$E_K(X_j) \triangleq X_j \oplus K_j, \quad \forall j \in \mathbb{Z}.$$

This is followed by compression, which is agnostic of K , to generate the compressed ciphertext $C(E_K(X))$.

The key insight underlying the approach in [4] is that the problem of compression in this case can be formulated as a Slepian-Wolf coding problem. In this formulation the ciphertext $E_K(X)$ is cast as a source, and the shared key K is cast as the decoder-only side-information. The joint distribution of the source and side-information can

¹Note, though, that only the encryptor has access to the key.

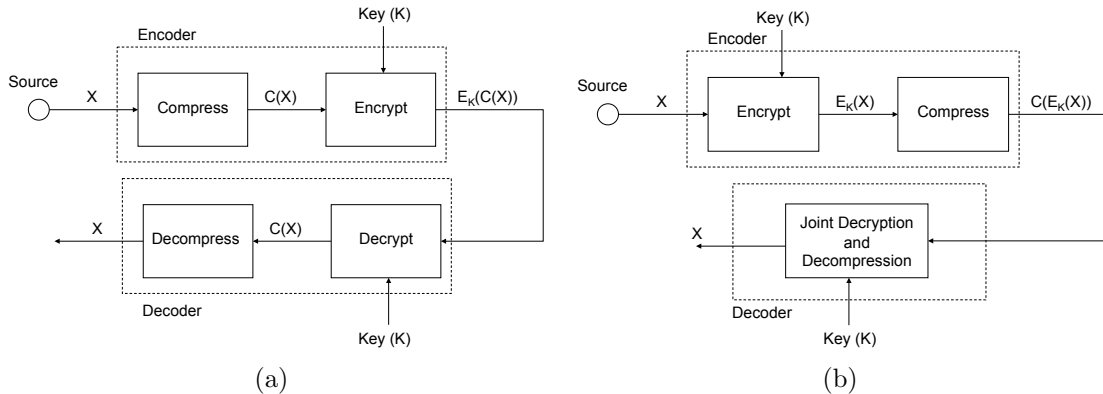


Figure 2: Systems combining compression and encryption. (a) Traditional system with compression done first. (b) System with encryption done before compression.

be determined from the statistics of the source. For example, in the binary case with a uniformly distributed K and X with $\Pr[X = 1] = p$, we have

$$P(E_K(X_j) \neq k | K = k) = p. \quad (1)$$

The decoder has knowledge of K , and of the source statistics. It uses this knowledge to reconstruct the ciphertext $E_K(X)$ from the compressed message $C(E_K(X))$, and to subsequently decrypt the plaintext X . This formulation is leveraged in [4] to show that exactly the same lossless compression rate, $H(X)$, can be asymptotically achieved in the system shown in Figure 2(b), as can be achieved in Figure 2(a). Further, this can be done while maintaining information-theoretic security.

The one-time pad and stream ciphers, while convenient for analysis, are not the only forms of encryption in practice. In fact, the prevalent method of encryption uses block ciphers in chaining modes such as CBC (see next section). Thus, an obviously desirable extension of the technique in [4] would be to conventional encryption schemes such as the popular AES encryption method. Attempting to do so, however, proves to be problematic. The method in [4] leverages the fact that in a one-time pad encryption scheme a simple symbol-wise correlation exists between the key K and the ciphertext $E_K(X)$, as seen in (1). Unfortunately, for block ciphers such as AES no such correlation structure is known. Moreover, any change in the plaintext is diffused in the ciphertext, and quantifying the correlation (or the joint probability distribution) of the key and the ciphertext is believed to be computationally infeasible and a requirement for the security of the block cipher.

In the rest of this paper, we will show how this problem can be circumvented by exploiting the chaining modes popularly used with block ciphers. Based on this insight, we will present an approach for compressing data encrypted with AES, without knowledge of the key. As in [4], the proposed method is based on the use of Slepian-Wolf coding.

3 Compressing Block-cipher Encryption

In contrast to stream ciphers, such as the one-time pad, block ciphers are highly nonlinear and the correlation between the key and the ciphertext is, by design, hard to characterize. Nevertheless, we proceed to show how this difficulty can be circumvented and how block ciphers can be compressed when they are used in conjunction with the popularly used chaining modes. We note that AES is used as an example of a block cipher, but our techniques apply to any block cipher.

If a block cipher operates on each block of data individually, two identical inputs will produce two identical outputs. While this weakness does not necessarily enable an unauthorized user to understand contents of an individual block it can give him information about frequently occurring data patterns. To address this problem, various chaining modes, also called modes of operation, are used in conjunction with block ciphers. The idea is to randomize each plaintext block, by using a randomization vector derived as a function of previous encryptor inputs or outputs. The randomization prevents two identical plaintext blocks from being encrypted into two identical ciphertext blocks, thus preventing leakage of information about data patterns.

We will primarily focus on the widely used CBC mode. The CBC mode of operation is depicted in Figure 3. We denote by X_i the i -th plaintext block. An initial

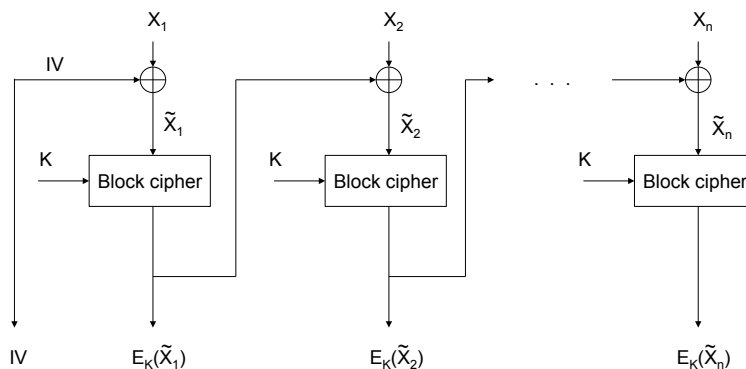


Figure 3: Cipher block chaining (CBC).

pseudorandom vector, used to initiate the chaining, is denoted as IV . Ciphertext $E_K(X)$ is generated by applying the AES encryption algorithm with key K to the plaintext X . In the CBC mode, each plaintext block X_i is randomized prior to encryption, by being XOR-ed with the ciphertext block corresponding to the previous plaintext block X_{i-1} , to obtain \tilde{X}_i . Thus, the i -th ciphertext block is generated as

$$E_K(\tilde{X}_i) = E_K(X_i \oplus E_K(\tilde{X}_{i-1})), \quad (2)$$

where the pseudorandom vector IV , assumed to be drawn uniformly from the source alphabet, is used instead of $E_K(\tilde{X}_0)$. This method of chaining ensures that frequently occurring plaintext patterns do not lead to repeating ciphertext patterns. Note that block ciphers in CBC mode are employed as the default mechanism in widespread security standards such as IPsec [6] and TLS/SSL [7] and hence it is the prevalent

method of encrypting internet traffic. For the remainder of this section, we will assume that the length of a plaintext block is m , and that X_i and \tilde{X}_i are drawn from the same binary extension field \mathcal{X}^m . We will further assume that X_i is generated by an i.i.d. source with marginal distribution P_X .

The key insight underlying the proposed approach can now be described. The statistical relationship between the key K and the i -th AES encrypted ciphertext $E_K(\tilde{X}_i)$ is hard to characterize. However, the joint distribution of the randomization vector $E_K(\tilde{X}_{i-1})$ and the i -th input to the AES encryptor \tilde{X}_i is easier to characterize, as it is governed by the distribution of the plaintext block X_i . For example, in the i.i.d source case we are considering, $E_K(\tilde{X}_{i-1})$ and \tilde{X}_i are related through a symbol-wise model governed by the distribution P_X . The correlation induced by the use of the chaining mode can be exploited to allow compression of encrypted data using Slepian-Wolf coding, as we will now show.

Let $\{C_{m,R}, D_{m,R}\}$ denote an order m Slepian-Wolf code with encoding rate R . Here, the Slepian-Wolf encoding function $C_{m,R}$ is a mapping from \mathcal{X}^m to the index set $\{1, \dots, 2^{mR}\}$, and the Slepian-Wolf decoding function $D_{m,R}$ is a mapping from $\{1, \dots, 2^{mR}\} \times \mathcal{X}^m$ to \mathcal{X}^m . The proposed compression method is illustrated in Figure 4. The input to the compressor consists of the IV and n ciphertext

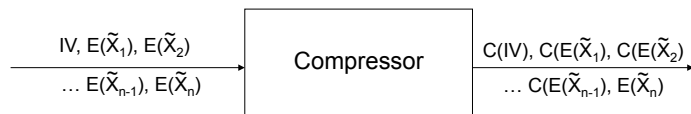


Figure 4: Compressor.

blocks $E_K(\tilde{X}_i)$, $1 \leq i \leq n$, generated by the AES cryptosystem running in the CBC mode. Since $E_K(\tilde{X}_i) \in \mathcal{X}^m$, the total length of the input sequence is $(n + 1) \cdot m \cdot \log |\mathcal{X}|$ bits. The compressor applies the Slepian-Wolf encoder $C_{m,R}$ to the IV and each of the first $n - 1$ ciphertext blocks independently, while the n -th block is left unchanged. Thus, the output of the compressor is the sequence $C(IV), C(E_K(\tilde{X}_1)), C(E_K(\tilde{X}_2)), \dots, C(E_K(\tilde{X}_{n-1})), E_K(\tilde{X}_n)$, where the subscripts are omitted for notational simplicity. The length of the output sequence is $n \cdot m \cdot R + m \cdot \log |\mathcal{X}|$ bits. Thus, the compressor achieves a compression factor of

$$\frac{(n + 1) \cdot m \cdot \log |\mathcal{X}|}{n \cdot m \cdot R + m \cdot \log |\mathcal{X}|} \approx \frac{\log |\mathcal{X}|}{R}$$

for large n . Note that the compressor does not need to know the key K . Also, note that this approach only requires a compressed IV, which by itself is incompressible, therefore no performance loss is inflicted by the uncompressed last block.

The joint decompression and decryption method is shown in Figure 5. The received compressed sequence is decrypted and decompressed serially, from right to left. In the first step $E_K(\tilde{X}_n)$, which is received without compression, is decrypted using the key K to generate \tilde{X}_n . Next, Slepian-Wolf decoding is performed to reconstruct $E_K(\tilde{X}_{n-1})$ using \tilde{X}_n as side-information, and the compressed bits $C(E_K(\tilde{X}_{n-1}))$. The decoder computes $\hat{E} \triangleq D(C(E_K(\tilde{X}_{n-1})), \tilde{X}_n)$, such that $\hat{E} = E_K(\tilde{X}_{n-1})$ with

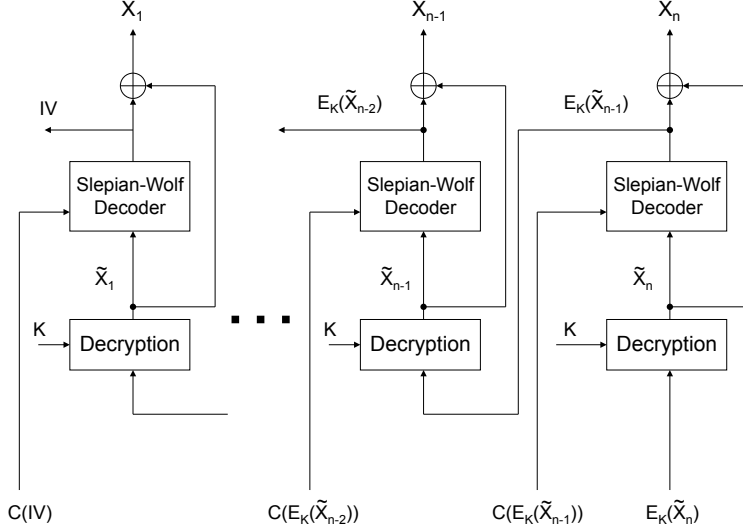


Figure 5: Joint decryption and decoding at the receiver. It is performed serially from right to left.

high-probability if the rate R is high enough. Once $E_K(\tilde{X}_{n-1})$ has been recovered by the Slepian-Wolf decoder, the plaintext block can now be reconstructed as $X_n = E_K(\tilde{X}_{n-1}) \oplus \tilde{X}_n$. The decoding process now proceeds serially with $E_K(\tilde{X}_{n-1})$ decrypted to generate \tilde{X}_{n-1} , which acts as the new Slepian-Wolf side-information. This continues until all plaintext blocks have been reconstructed.

For large m , it follows from the Slepian-Wolf theorem that the rate required to ensure correct reconstruction of the $(i-1)$ -th block with high probability is given as

$$\begin{aligned}
 R &= H(E_K(\tilde{X}_{i-1})|\tilde{X}_i) = H(E_K(\tilde{X}_{i-1})|E_K(\tilde{X}_{i-1}) \oplus X_i) \\
 &\leq H(E_K(\tilde{X}_{i-1}) \oplus X_i|E_K(\tilde{X}_{i-1})) = H(X_i).
 \end{aligned} \tag{3}$$

If we assume that $E_K(\tilde{X}_{i-1})$ has a uniform distribution, (3) becomes an equality. In practice, as we will see in Section 4, m is typically small. In this case, the required rate R is a function of P_X , m , the acceptable decoding error probability, and the non-ideal Slepian-Wolf codes used.

The above description focuses on the CBC mode as the most common form of encryption, but our techniques can be extended to other CBC-like modes of operation.

We note that it can be formally proved that the compression and decompression operations that we introduced on top of the regular CBC mode do not compromise the security of the original CBC encryption. The proof follows standard techniques in the cryptographic literature and uses the fact that both compression and decompression algorithms access the encryption and decryption functionalities as black boxes (rather than utilizing the internals of the block cipher). Further, it can be shown that there exists a fundamental limitation to the compression capability when the input to the block cipher is a single-block message (no chaining, no IV). This limitation, expressed in the form of a computational lower bound on any effective compression algorithm

that works for any block cipher on single-block ciphertexts, indicates that the gains in compression that we obtained for CBC mode really come from the chaining operation and not from the (efficient) compressibility of single ciphertext blocks. This computational lower-bound argument uses in an essential way the security definition of block ciphers, and thus represents an interesting tradeoff between cryptographic security and the computational hardness of ciphertext compression. Full proofs of these claims are omitted here due to lack of space, but we plan to present them in the full version of this paper.

4 Compression Performance

The efficiency of Slepian-Wolf compression depends on the performance of underlying Slepian-Wolf codes. It was shown in [12, 13] that Slepian-Wolf compression approaches entropy with speed $O(\sqrt{\frac{\log n}{n}})$, which is considerably slower than $O(\frac{1}{n})$ of arithmetic coding. It follows that for efficient Slepian-Wolf compression, the block length of Slepian-Wolf codes must be long.

Codes that are used to compress stream ciphers as described in Section 2 can be chosen to have arbitrary block lengths, since the method itself does not directly impose any constraints on the block length. On the other hand, our method for compressing block ciphers must operate block-wise since decompression occurs serially and depends upon a previously decompressed block. In effect, the block length of Slepian-Wolf codes must be m . Like in the previous section we choose AES as a representative of a widely used block cipher and present our results under assumption that AES is used as the encryption scheme.

Slepian-Wolf codes over finite block lengths have nonzero frame-error rates (FER), which implies that the receiver will sometimes fail to recover $E_K(\tilde{X}_{i-1})$ correctly. Such errors must be dealt with on the system level, as they can have catastrophic consequences in the sense that they propagate to all subsequent blocks. In the following it is assumed that as long as the FER is low enough, the system can recover efficiently, for instance by supplying the uncompressed version of the erroneous block to the receiver. The compression performance depends on the target FER.

We consider a binary i.i.d source with a probability distribution $\Pr(X = 1) = p$ and $\Pr(X = 0) = 1 - p$ which produces plaintext bits. A sequence of plaintext bits is divided into blocks of size m , which are encrypted and compressed as described in Section 3. The receiver's task is to reconstruct $E_K(\tilde{X}_{i-1})$ from \tilde{X}_i and side information $C(E_K(\tilde{X}_{i-1}))$. For the source considered, Slepian-Wolf decoding is equivalent to error-correction over a binary symmetric channel (BSC), thus the underlying codes should yield a FER that is lower or equal to the target FER over the BSC. Compression efficiency can be evaluated in two ways:

- a) fix p and determine the compression rate of a Slepian-Wolf code that satisfies the target FER;
- b) pick a well-performing Slepian-Wolf code and determine the maximum p for which target FER is satisfied.

We use low-density parity-check (LDPC) codes [14], which are known to be very powerful and we evaluate compression performance according to method b). In our simulations we used two LDPC codes²: the first yields compression rate 0.5 and has degree distribution $\lambda(x) = 0.3317x + 0.2376x^2 + 0.4307x^5$, $\rho(x) = 0.6535x^5 + 0.3465x^6$ and the second yields compression rate 0.75 and degree distribution $\lambda(x) = 0.4249x + 0.0311x^2 + 0.5440x^4$, $\rho(x) = 0.8187x^3 + 0.1813x^4$. All codes were constructed with the Progressive Edge Growth algorithm [15], which is known to yield good performance at short block lengths. Belief propagation [14] is used for decoding and the maximum number of iterations was set to 100.

p	Target FER	Compression Rate	Source Entropy
0.026	10^{-3}	0.50	0.1739
0.018	10^{-4}	0.50	0.1301
0.068	10^{-3}	0.75	0.3584
0.054	10^{-4}	0.75	0.3032

Table 1: Attainable compression rates for $m = 128$ bits.

Simulation results for block length, m , 128 and 1024 bits are shown in Tables 1 and 2, respectively. First, consider the current specification of the AES standard [2], where m is 128 bits. At FER of 10^{-3} the maximum p that can be compressed to rate 0.5 is 0.026, while the entropy for that p is 0.1739. The large gap is a consequence of a very short block length. Notice that for higher reliability, i.e. lower FER, the constraints on the source are more stringent for a fixed compression rate.

p	Target FER	Compression Rate	Source Entropy
0.058	10^{-3}	0.50	0.3195
0.048	10^{-4}	0.50	0.2778
0.134	10^{-3}	0.75	0.5710
0.126	10^{-4}	0.75	0.5464

Table 2: Attainable compression rates for $m = 1024$ bits.

When the m is increased to 1024 bits (see Table 2) the improvement in performance is considerable. For instance, at FER = 10^{-3} and compression rate 0.5, the source can now have p up to 0.058. In future block-cipher designs one could consider larger blocks, in particular as a way to allow for better post-encryption compression.

5 Conclusion

We considered compression of data encrypted with block ciphers without knowledge of the key. Contrary to a popular belief that such data is practically incompressible we show how compression can be attained for binary sources. Our method is

²The degree distributions were obtained at <http://lthcwww.epfl.ch/research/ldpcopt/>

based on Slepian-Wolf coding and hinges on the fact that chaining modes, which are widely used in conjunction with block ciphers, introduce simple symbol-wise correlation between successive blocks of data. Simulation results indicate that, while still far from theoretical limits, considerable compression gains are attainable and improved performance can be expected as block sizes increase in the future.

References

- [1] G. S. Vernam, “Cipher printing telegraph systems for secret wire and radio telegraphic communications,” *Journal of the American Institute for Electrical Engineers*, vol. 55, pp. 109–115, 1926.
- [2] W. Mao, *Modern Cryptography: Theory and Practice*. Prentice Hall, 2003.
- [3] N. B. of Standards, *Data Encryption Standard (DES)*. U.S. Department of Commerce, Washington D.C, 1977.
- [4] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, “On compressing encrypted data,” *IEEE Trans. Signal Processing*, vol. 52, pp. 2992–3006, Oct. 2004.
- [5] D. Slepian and J. Wolf, “Noiseless coding of correlated information sources,” *IEEE Trans. Info. Theory*, vol. 19, pp. 471–480, July 1973.
- [6] S. Kent and K. Seo, “Security achitecture for the internet protocol,” in *RFC 4301*, Dec. 2005.
- [7] T. Dierks and E. Rescorla, “The tls protocol – version 1.2,” in *RFC 5246*, Aug. 2008.
- [8] M. Johnson, D. Wagner, and K. Ramchandran, “On compressing encrypted data without the encryption key,” in *Proc. of the Theory of Crypto. Conf.*, Cambridge, MA, Feb. 2004.
- [9] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.
- [10] A. Aaron and B. Girod, “Compression with side information using turbo codes,” in *IEEE Data Compression Conf.*, 2002, pp. 252–261.
- [11] J. Garcia-Frias, “Compression of correlated binary sources using turbo codes,” *IEEE Communications Letters*, vol. 5, pp. 417–419, Oct. 2001.
- [12] D. He, L. Lastras-Montaña, and E. Yang, “A lower bound for variable rate slepian-wolf coding,” in *IEEE Inter. Symp. on Info. Theory*, Seattle, WA, July 2006.
- [13] —, “On the relationship between redundancy and decoding error in slepian-wolf coding,” in *IEEE Information Theory Workshop*, Chengdu, China, Oct. 2006.
- [14] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.
- [15] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, “Regular and irregular progressive edge-growth tanner graphs,” *IEEE Trans. Info. Theory*, vol. 51, pp. 386–398, Jan. 2005.