On Compressing Encrypted Data Without the Encryption Key *

Mark Johnson, David Wagner, and Kannan Ramchandran

Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley, CA 94720, USA.
e-mail: {mjohnson, kannanr}@eecs.berkeley.edu, daw@cs.berkeley.edu

Abstract. When it is desired to transmit redundant data over an insecure and bandwidth-constrained channel, it is customary to first compress the redundant data and then encrypt it for security reasons. In this paper, we investigate the novelty of reversing the order of these steps, i.e. first encrypting and then compressing. Although counter-intuitive, we show surprisingly that through the use of coding with side information principles, this reversal in order is indeed possible. In fact, for lossless compression, we show that the theoretical compression gain is unchanged by performing encryption before compression. We show that the cryptographic security of the reversed system is directly related to the strength of the key generator.

1 Introduction

Consider the problem of transmitting redundant data over an insecure, bandwidth-constrained communications channel. It is desirable to both compress and encrypt the data. The traditional way to do this is to first compress the data to strip it of its redundancy followed by an encryption of the compressed bitstream. In this paper, we investigate the novelty of reversing the order of these steps, i.e. first encrypting and then compressing, and the effect of that reversal on the compression efficiency and the cryptographic security.

We present a scheme, based on distributed source coding, that enables us to realize this reversal of operations. Our scheme allows us to compress a stationary, i.i.d. source that has been encrypted with a stream cipher (cf., [1]) to a rate close to the entropy rate of the source. Although the code that is used to compress the encrypted source is entirely different from the code that would be used to compress the original source, we can in fact compress the encrypted source to the same rate as we could have compressed the original source. We focus exclusively on this class of stationary, i.i.d. sources in this work. The existence of linear codes that achieve these compression gains can be proven in a non-constructive manner. Furthermore, recent results from distributed source coding can be applied to this problem to give constructions for codes that can compress

* This research was supported by the Fannie and John Hertz Foundation and by NSF under grants CCR-0093337, CCR-0325311, and CCR-0208883.

any i.i.d. source that has been encrypted with a stream cipher. In general, these codes will have inefficient decoding algorithms, which limits their usefulness. However, for the case of binary i.i.d. sources, we present code constructions from the literature that support computationally efficient decoding and still achieve compression gains close to the information-theoretic bounds.

Our scheme requires that the decompression algorithm have access to the cryptographic key, but importantly, the compression algorithm does not receive the key. The compressor must know the entropy rate of the original source in order to select an appropriate code, but it does not use the encryption key. To be specific, the compressor only needs to know the entropy rate of the source, not the full distribution. Our scheme is statistical in nature, and there is the possibility that the output of the decoder will not match the original source. We show that for i.i.d. sources this probability of error decreases exponentially toward 0 as the blocklength of the code increases.

While we focus here on the theoretical feasibility of our claim, we have uncovered a few application scenarios of possible interest. In one scenario, the generator of the redundant data (the content author) has no incentive to compress the data as it is not interested in saving bandwidth that it does not own at the cost of unnecessary computational complexity. Nevertheless, the content generator is very interested in protecting the privacy of the content via encryption. This content is typically distributed to its client base by a content distribution unit which has great incentive to remove all redundancy from the content in order to maximize its network utilization. However, there is no trust between the content generator and the compressor, so the former will supply only encrypted data to the latter. Our scheme allows the compressing unit to compress the encrypted data at the same efficiency as if it was compressing the original, unencrypted data, even though the compressor does not have access to the key used in the encryption step.

The main contribution of this work is in the identification of the connection between the stated problem and distributed source coding, as well as an analysis of the compression efficiency and cryptographic security of our scheme. This paper is organized as follows. Section 2 gives some background information. The scheme for compressing encrypted data is presented in Section 3. The cryptographic security of the scheme is studied in Section 4. Related work is discussed in Section 5. Some conclusions and future work are described in Section 6.

2 Background

Before describing our solution to the problem of compressing encrypted data, we will briefly present some background information. First, we will discuss the principles of distributed compression that underpin our solution. Then we will cover some concepts from cryptography that will be used to quantify the strength of the encryption.

2.1 Distributed Source Coding

Distributed source coding considers the problem of separately compressing sources X and S that are correlated, where the two compressors cannot communicate with each other. The Slepian-Wolf theorem [2] gives the smallest rates required to losslessly communicate X and S to the decoder, when both X and S come from memoryless sources outputting an unending stream of i.i.d. values. The Slepian-Wolf theorem is a non-constructive result that states these smallest rates, but does not show how to construct codes that approach the minimum rates. For a practical code construction there will be a tradeoff between the blocklength and the probability or error, i.e. as the blocklength increases the probability of error can be made smaller. However, this theorem also does not provide any specific insight what the tradeoff is. Subsequent work by Csiszar [3], which we discuss in Theorem 1 in Section 3, has shown that linear codes can approach the bounds given by the Slepian-Wolf theorem.

An important special case of this problem, upon which we will focus, is when X needs to be sent to a decoder which has access to the correlated side-information S. For this special case, the Slepian-Wolf theorem asserts that the minimum rate required to transmit X is given by the conditional entropy (cf., [4]) of X given S, denoted by $H(X|S)$ bits/sample.

While the Slepian-Wolf theorem is non-constructive, there has been some recent work that provides practical code constructions to realize these distributed compression gains [5]. We will use an example to show the intuition behind these constructions.

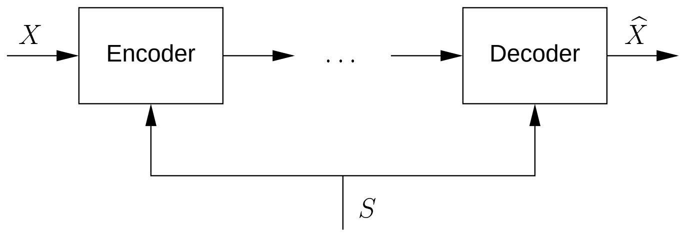
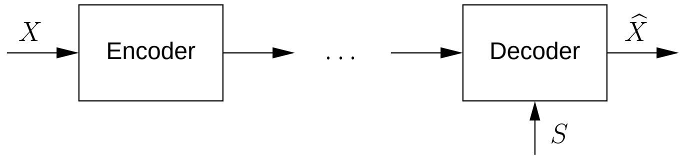


Fig. 1. A source coding with side information problem: *The side information S is available at both the encoder and the decoder.*

We begin by looking at the problem where S is available at both the encoder and the decoder, as depicted in Figure 1. In our example, X and S are correlated, uniformly distributed binary strings of length 3. The correlation structure is such that their Hamming distance is at most 1, i.e. they differ in at most one of the three bit positions. For example, if X is 010, then S will equally likely be one of the four patterns {010,011,000,110}. The encoder forms the error pattern $e = X \oplus S$. Because X and S differ in at most one bit position, the error pattern e can take on only four possible values, namely {000,001,010,100}. These four values can be indexed with two bits. That index is transmitted to the decoder, which looks up the error pattern corresponding to the index received from the encoder, and then computes $X = e \oplus S$.



$$\underset{(00)}{\text{Coset 1}} = \begin{Bmatrix} 0\;0\;0 \\ 1\;1\;1 \end{Bmatrix} \quad \underset{(01)}{\text{Coset 2}} = \begin{Bmatrix} 0\;0\;1 \\ 1\;1\;0 \end{Bmatrix}$$

$$\underset{(10)}{\text{Coset 3}} = \begin{Bmatrix} 0\;1\;0 \\ 1\;0\;1 \end{Bmatrix} \quad \underset{(11)}{\text{Coset 4}} = \begin{Bmatrix} 1\;0\;0 \\ 0\;1\;1 \end{Bmatrix}$$

Fig. 2. A source coding with side information problem: *X and S are three bit binary sequences which differ by at most one bit. S is available only at the decoder. The encoder can compress X to two bits by sending the index of the coset in which X occurs.*

Now, we consider the case in Figure 2 where S is available at the decoder, but not the encoder. Without S, the encoder cannot form the error pattern e. However, it is still possible for the encoder to compress X to two bits and for the decoder to reconstruct X without error. The reason behind this surprising fact is that there is no reason for the encoder to spend any bits to differentiate between $X = 000$ and $X = 111$. The Hamming distance of 3 between these two codewords is sufficiently large to enable the decoder to correctly decode X based on its access to S and the knowledge that S is within a Hamming distance of 1 from X. If the decoder knows X to be either $X = 000$ or $X = 111$, it can resolve this ambiguity by checking which of the two is closer in Hamming distance to S, and declare that codeword to be X. We observe that the set {000,111} is a 3-bit repetition code with a Hamming distance of 3.

Likewise, in addition to the set {000,111}, we can consider the following 3 sets for X: {100,011}, {010,101}, and {001,110}. Each of these sets is composed of two codewords whose Hamming distance is 3. These sets are the cosets of the 3-bit repetition code. While we typically use the set {000,111} as the 3-bit repetition code (0 is encoded as 000, and 1 as 111), it is clear that one could just as easily have used any of the other three cosets with the same performance. Also, these 4 sets cover the complete space of binary 3-tuples that X can assume. Thus, instead of describing X by its 3-bit value, all we need to do is to encode the coset in which X occurs. There are 4 cosets, so we need only 2 bits to index the cosets. We can compress X to 2 bits, just as in the case where S was available at both the encoder and decoder.

In practical situations, the correlation structure between X and S is often not as simple as in this example. For instance, X and S could be three-bit binary numbers such that the Hamming distance between X and S is equal to 0 or 1 with probability $1 - 10^{-6}$. If we compress X with the same code construction as above, then with probability $1 - 10^{-6}$ the Hamming distance between X and S will be at most 1 and $\hat{X}$ will be equal to X. However, with probability 10^{-6} the Hamming distance between X and S will be more than 1. In that case, $\hat{X}$ and X will be different, which means that the decoder has incorrectly decoded the message. The important point is that in these code constructions, unlike in standard source codes, there is a probability of error at the decoder.

In practice, we will use a much more complex channel code than the simple repetition code. The channel code is chosen based on the correlation structure between X and S, so as to minimize the probability of error. However, the encoding and decoding procedures are the same as in our three-bit example. The encoder finds the coset which contains X and transmits the index of this coset. The decoder finds the codeword in the coset denoted by the received index which is closest to S in the Hamming metric.

2.2 Security

We will express an arbitrary encryption scheme with the notation $c = E_k(m)$. Here, m is the plaintext, c is the ciphertext, and k is the key used by the algorithm.

We will quantify the security of an encryption scheme against chosen-plaintext attacks by means of the concept of left-or-right (LOR) security, which was introduced in [6]. The central feature of LOR security is an oracle which supplies responses to queries. A query consists of a pair of plaintexts, denoted by (x, y). The response of the oracle will be the encryption of one of the two plaintexts in the query. There are two types of oracles. A left oracle, which we denote by $\S_0$, will always return the encryption of the first plaintext in the query. The functionality of selecting the first plaintext is denoted by the function $\S_0$, where we define $\S_0(x, y) = x$. In contrast, the right oracle uses a selection function, written $\S_1$, which always returns its second argument: $\S_1(x, y) = y$. In either case, the result of the selection algorithm is encrypted using E_k. Consequently, the functionality of the left oracle can be expressed as $E_k \circ \S_0$, and the right oracle as $E_k \circ \S_1$.

In the left-or-right security model, one of the two types of oracles is chosen at random. An adversary, denoted by A, attempts to determine whether the oracle is a left oracle or a right oracle by making queries to it. Intuitively, if the encryption scheme is very weak then the adversary will be able to examine a ciphertext response and determine with high probability which of the two plaintexts in the query was encrypted. Conversely, for a very strong encryption scheme it is difficult to match plaintexts to the corresponding ciphertexts: the adversary cannot do much better than randomly picking one of the two plaintexts.

We use the superscript notation $A^{E_k \circ \S_0}$ to denote the output of the adversary after interacting with a left oracle, and $A^{E_k \circ \S_1}$ for its result after interacting with a right oracle. The output of the adversary will be 0 if the adversary decides that the oracle is a left oracle, and 1 if the adversary decides that it is a right oracle.

For left-or-right security, we require that:

$$|\Pr[A^{E_k \circ \S_1} = 1] - \Pr[A^{E_k \circ \S_0} = 1]| \leq \varepsilon \tag{1}$$

In this equation, the probability is taken across the distribution of keys k and any randomness in the adversary. If this equation holds for all possible adversaries that run in time t and make q or fewer queries to the oracle, then we say that the encryption scheme E is (t, q, ε)-LOR-secure.

One other cryptographic concept which we will use is the variational distance. This distance measures the dissimilarity between two probability distributions [1]. If D_1 and D_2 are two probability distributions, then the variational distance $V(D_1, D_2)$ between the two is defined as:

$$V(D_1, D_2) = \sum_x |\Pr_{D_1}[x] - \Pr_{D_2}[x]|. \tag{2}$$

3 Secure Compression Scheme

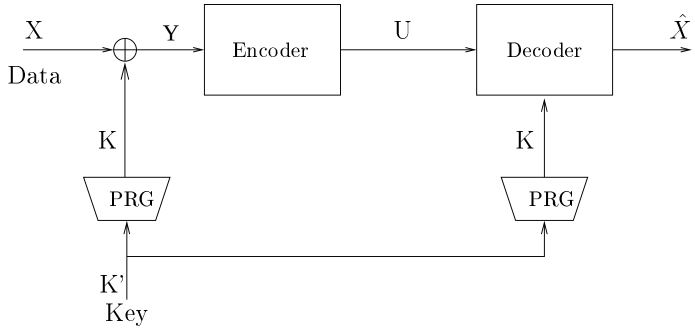


Fig. 3. Secure compression scheme: *The data X is first encrypted with a stream cipher and then compressed. At the receiver, decoding and decryption are performed in a joint step. The eavesdropper sees U, but (as we show) cannot learn anything useful about X.*

Our scheme for compressing encrypted data is illustrated in Figure 3. The message is first encrypted with a binary stream cipher. The seed K' is used as the input to a pseudo-random generator (PRG), whose output is denoted by K. The message is encrypted by forming the bitwise binary sum $Y = X \oplus K$. Then, Y is compressed to obtain the result $U = C(Y)$, and U is transmitted to the

[1] The variational distance is related, but not identical, to the K-L divergence (cf., [4]).

recipient. The adversary is assumed to be able to eavesdrop on the ciphertext U.

Throughout this work, we will assume that all sources are memoryless. In other words, we have a distribution D on some alphabet $\mathcal{X}$; the source outputs an unending sequence of i.i.d. random variables, each distributed according to D. If X denotes the sequence of outputs from such a source, we will sometimes write X_1, X_2, etc., for the first, second, etc., output from the source. Also, if X is such a source, we write X^n for the source that outputs a block of n items from X at a time. The first output of X^n is $(X_1, X_2, \ldots, X_n)$ (which is distributed according to $D \times \cdots \times D$); the second output of X^n is $(X_{n+1}, \ldots, X_{2n})$; and so on. The entropy rate of the source X is $H(D)$, or sometimes written just $H(X)$; consequently, the entropy rate of X^n is $n \cdot H(X)$.

Also, we assume that the seed has been transmitted to the decoder through a secure channel. By implementing an identical PRG, the decoder also has access to K. This reduces the problem of compressing encrypted data to the distributed source coding problem. Y and K are two correlated sources, because Y was generated via $Y = X \oplus K$. Our goal is to compress Y, using the fact that K is available at the decoder as side information. The Slepian-Wolf theorem asserts that Y can be compressed to a rate of $H(Y|K)$. Because $Y = X \oplus K$, it follows that $H(Y|K) = H(X)$. Hence, we can compress the encrypted source Y to a rate $H(X)$, which is the entropy rate of the original source X, conditioned on the fact that K is available at the decoder. In order to select a code to perform this compression, the encoder needs only to know $H(X)$, not the full distribution on X.

3.1 Theoretical Bounds on Compression Efficiency

Although the Slepian-Wolf theorem is non-constructive, the distributed source coding using syndromes (DISCUS) framework [5] provides a constructive method of achieving the coding gains promised by Slepian-Wolf through the use of linear codes. In order for the DISCUS encoder to select a code matched to the particular source, it only needs to know the entropy rate of the original source, not the full distribution. Csiszar showed in [3] that linear codes can achieve the bounds given by the Slepian-Wolf theorem as the block length of the code approaches ∞. Therefore, by restricting our attention to linear codes we do not reduce the compression gains that can be achieved. We present the following theorem, which is our statement of a more general theorem from Csiszar as applied to our specific problem.

Theorem 1 (Csiszar). *Let X and Y be memoryless sources with entropy $H(X)$ and $H(Y)$, respectively. Suppose that X and Y are correlated, in the following sense: we have a joint distribution function $f(x, y)$ which describes the distribution of (X_i, Y_i) for every i. Also, suppose that Y^n is side-information that is available perfectly at the decoder. Then, for every $\epsilon > 0$ and every blocklength n, there exists a* linear *code C_n for compressing the source X^n at an encoding rate of $n \cdot (H(X|Y) + \epsilon)$ such that the probability p_e of not being able to recover the source message correctly at the decoder is bounded above by: $p_e \leq \exp\{-n \cdot g(\epsilon) + o(n)\}$.*

The function $g(\cdot)$ depends on the distribution (X, Y), but not on n. Thus, Csiszar's work shows not only that the probability of error p_e can be made arbitrarily small for large blocklengths and rates greater than $H(X|Y)$, but also that p_e will decrease exponentially with the blocklength n for suitably chosen codes. Although this theorem proves the existence of linear codes that approach the Slepian-Wolf bounds, it makes no guarantees on the decoding complexity or uniformity of such codes. In general, the codes will require computationally inefficient decoding and nonuniform encoding.

Notice that Csiszar's result is also non-constructive, in that it proves only the existence of a linear code achieving these rates, but does not specify how to find such a linear code explicitly. Hence, this result is achieved in a non-uniform model of computation: the code might depend in a non-uniform way on ϵ, p, X, Y, f, and n; there is no guarantee that we can find a single compression algorithm that works for all n. These non-constructive aspects are unfortunate, but we do not know how to avoid them.

The advantage of using a linear code is in the computational complexity required to implement the encoder. The encoder divides the encrypted source Y into blocks of length n. Each block, which we will denote as Y_i^n, is then mapped to a value U_i in the set $\{0, 1\}^{nR}$, where R is the rate of the code. Hence, U_i can be represented by nR binary digits. This mapping is performed via a simple matrix multiplication, $U_i = H^T Y_i^n$, where H is a matrix of size nR by n that is referred to as a parity check matrix in the coding theory literature (cf., [7])[2]. The complexity of the compressor is quadratic in the block length, since the encoder compresses each block by performing a single matrix multiplication.

The parity check matrix used in the encoder corresponds to a particular code. This code in turn partitions the space of all n bit binary numbers into cosets, just as in the example in Section 2 where the repetition code partitioned the space of three-bit binary numbers into four cosets. The decoder finds the codeword in the coset indexed by U_i which is closest to the side information K_i^n in Hamming distance. This codeword is denoted by $\hat{Y}_i^n$. The decoder then computes its estimate of the original source X_i, which is $\hat{X}_i^n = \hat{Y}_i^n \oplus K_i^n$.

Unfortunately, the complexity of the decoder is not as easily quantified as the encoder, and is highly dependent on the particular code used. We don't know the best achievable time for decoding in Csiszar's result, but we suspect that it may be exponential in n. For instance, we could find the $\hat{Y}_i^n$ which is the maximum likelihood estimate of Y_i^n by exhaustive search through all of the codewords in the coset indexed by U_i, but the complexity of such a search would be exponential in n.

Also, the super-polynomial complexity of the decoder raises a correctness issue for compressing encrypted data: There is, in general, no guarantee that the decoder's error probability remains bounded by about p_e in Figure 3, since the PRG is not guaranteed to remain pseudorandom against distinguishers with

[2] This can be readily generalized to non-binary sources. The elements of U and H are then from the same field as X.

super-polynomial running time. However, it seems reasonable[3] to assume we can find a PRG whose security when used with κ-bit keys is $\exp\{\Omega(\kappa^{\delta})\}$, for some constant $\delta > 0$. In this case, we can choose parameters so that the PRG provides security against all distinguishers running in time $|\mathcal{X}|^n$, say, and then we will obtain a polynomial-time encoder and a decoder whose error probability is bounded by not much more than p_e.

In summary: Csiszar's result assures us of the existence of linear codes that provably meet the Slepian-Wolf rate, if our sources are memoryless. The encoder runs in polynomial time, but the decoder is suspected to require exponential time. (Note that our security results do not make any assumptions about the running time of the decoder; thus, our security claims for stream cipher encryption followed by Csiszar-style encoding are fully proven, not heuristic.) Also, Csiszar's result is non-constructive. Consequently, though Csiszar's results suffice to show (in principle) that compression can be securely and efficiently performed after encryption, the scheme thus obtained is, in several respects, not very attractive in practice.

3.2 Efficient Code Constructions

For practical uses, we have candidate compression schemes that seem to behave much better. We outline next several such codes that seem to have reasonable encoding and decoding complexity (in particular, with polynomial running time) and that seem to come very close to the Slepian-Wolf bound. In each case, we have extensive empirical evidence that these schemes behave well, but no theoretically proven guarantees. In practice, one would probably use one of these schemes.

The topic of linear codes (without side information) has been heavily studied in the coding literature, and several schemes are known that have computationally efficient, sub-optimal decoding algorithms. Turbo codes [8] and low density parity check (LDPC) codes [9] are two well-known examples. A class of LDPC codes known as expander codes were presented in [10] that were proven to have a polynomial-time decoding and to remove a constant fraction of the errors in a received codeword. Empirical results have consistently shown that LDPC codes have even better decoding performance than the proven bounds.

Recently, a significant amount of work has focused on applying both LDPC codes [11, 12] and turbo codes [13, 14] to the problem of source coding with side information. The authors of [12] consider a problem where the Slepian-Wolf theorem gives a bound of 0.466 bits/sample on the rate of the encoder. They used an LDPC code with block length 10^5 to compress the binary source to a rate of 0.5 bits/sample and had a probability of error at the decoder that was less than 10^{-6}. These schemes are constructive, and they have computationally efficient encoding and decoding routines. Since we have shown that the problem of compressing encrypted data is an example of source coding with side information,

[3] The existence of such a PRG is not guaranteed by the existence of one-way functions with super-polynomial security, but the usual candidate constructions of PRG's all seem to achieve this security level.

we could obtain the same compression gains in our problem by using the same code. Thus, in practice, one would probably use one of these schemes.

Using these constructions, we can efficiently compress any i.i.d. binary source that has been encrypted with a binary stream cipher. In theory, the encrypted source can be compressed to the entropy rate of the original source. In practice, by using LDPC or turbo codes, we can compress the encrypted source to a rate very close to the theoretical limit with a low probability of error. Obviously, if X is a Bernoulli(0.5) binary source, then it has an entropy rate of 1 bit/sample and we cannot compress the encrypted source. For any i.i.d. binary source with redundancy, however, we can compress Y to a rate close to $H(X)$. The gap between the rate of the encoded data and $H(X)$ is limited only by the code used in the encoder and decoder. In theory, a non-binary i.i.d. source could also be compressed to $H(X)$. Constructing codes for non-binary sources is a problem that has not been studied thoroughly, but if such codes were constructed then they could be used in this problem.

Note that in real-time applications, the use of block codes could prove problematic. Because the compressor cannot produce any output until it obtains n plaintext symbols, these approaches might add latency to the cryptosystem. The amount of latency will depend on the block length required to achieve the desired compression rate.

3.3 Other Encryption Algorithms

Up to this point, we have considered only a stream cipher encryption scheme. In a more general case, we could imagine using any encryption method whatsoever. For a general encryption method, it is still theoretically possible to compress the encrypted source to the entropy rate of the original source. However, in this case Y and K will have a very complex, non-linear correlation structure, whereas with a stream cipher the correlation between Y and K was the linear relationship $Y = K \oplus X$. Because the correlation structure is now nonlinear, we can no longer leverage existing channel code constructions to construct distributed source codes. The source coding with side information problem becomes much more difficult with a nonlinear correlation structure, and is not well studied in the coding theory literature.

The anonymous reviewers have pointed out that, in some cases, it may be possible to adapt our techniques to other encryption algorithms. If E is a secure encryption algorithm, then $E'(X) = (E(K'), G(K') \oplus X)$ is also secure (where K' is a fresh session key, randomly chosen for each message to be encrypted), and the second component of $E'(X)$ could be compressed using our scheme.

4 Cryptographic Security

In this section, we provide an analysis of the cryptographic security of the encryption step of our system, and we give a proof of security under plausible assumptions. In brief, the intuition behind our analysis is as follows: First, no computationally bounded attacker who observes Y can learn anything interesting about X, if the pseudorandom generator (PRG) is secure. Second, because U is computable from Y in polynomial time, no computationally bounded attacker who observes U can learn anything useful about X. Since U is what is actually transmitted across the insecure link, this will demonstrate that the system is a secure encryption scheme.

We will study the cryptographic security of the system in two steps. First, we will look at the case where the keystream K has been replaced with a Bernoulli(0.5) random variable. Then, we will extend this analysis to the case where K is the output of a PRG. Our main results will be stated as two theorems.

We begin with the information-theoretic case, where K is truly random. If the keystream K is truly random, then the stream cipher becomes a one-time pad scheme, and security will follow easily. In particular, let K be a source providing an unending stream of i.i.d. values uniformly distributed on $\{0,1\}^k$, chosen anew for each message transmitted independently of everything else (and assumed to be synchronized with the receiver). Let $C : \{0,1\}^k \to \{0,1\}^*$ be a compression algorithm, and define the cryptosystem $E_K : \{0,1\}^k \to \{0,1\}^*$ by $E_K(X) = C(X \oplus K)$ (more precisely, $E_{K_i}(X_i) = C(X_i \oplus K_i)$).

Theorem 2. *When K is uniformly distributed, $E_K(X) = C(X \oplus K)$ is $(\infty, q, 0)$-LOR secure.*

Proof. Fix any x, x', and let K be uniform. The distribution $x \oplus K$ is identical to the distribution $x' \oplus K$, hence $C(x \oplus K)$ has exactly the same distribution as $C(x' \oplus K)$. This means that, no matter the distribution of the random variables X, X', $(E_K \circ \S_0)(X, X')$ will have the same distribution as $(E_K \circ \S_1)(X, X')$. Consequently,

$$|\Pr[A^{E_K \circ \S_1} = 1] - \Pr[A^{E_K \circ \S_0} = 1]| = 0$$

for all adversaries A that make at most one query to their oracle. So, when $q = 1$, the scheme is $(\infty, 1, 0)$-LOR secure. For $q > 1$, we can use a straightforward hybrid argument. Consider the following hybrids, representing how each of the q oracle queries will be answered:

hybrid 0: $E_{K_1} \circ \S_0, E_{K_2} \circ \S_0, \ldots, E_{K_n} \circ \S_0$
hybrid 1: $E_{K_1} \circ \S_1, E_{K_2} \circ \S_0, \ldots, E_{K_n} \circ \S_0$

$\vdots$

hybrid q: $E_{K_1} \circ \S_1, E_{K_2} \circ \S_1, \ldots, E_{K_n} \circ \S_1$

By the above argument, A's output when run with hybrid i has the same distribution as A's output when run with hybrid $i+1$ (here we use that the value of K is chosen anew for each oracle query, independently of everything else). After a simple induction on q, we see that E_K is $(\infty, q, 0)$-LOR secure. □

Next, we consider the full scheme, where the keystream K is generated using a PRG. More specifically, let the secret key K' be distributed uniformly on $\{0,1\}^{k'}$, let $G : \{0,1\}^{k'} \to \{0,1\}^{kq}$ be a pseudorandom generator (PRG), and define $K = G(K')$. As before, let $C : \{0,1\}^k \to \{0,1\}^*$ be a compression algorithm.

This time, we will need to assume that the compression algorithm C runs in polynomial time. Finally, define the cryptosystem $E'_{K'} : \{0,1\}^k \to \{0,1\}^*$ by $E'_{K'}(X) = E_{G(K')}(X) = C(X \oplus G(K'))$; more precisely, $E'_{K'}(X_i) = C(X_i \oplus K_i)$, where $K = G(K')$. When this cryptosystem is used to encrypt multiple messages, we assume that we use consecutive outputs from the PRG, throwing away keystream bits after they are used. Of course, the encryptor must remember his place in the PRG output stream; hence, this is a stateful encryption scheme, and the sender and receiver must be synchronized. We now analyze the security of this cryptosystem.

Theorem 3. *Let $G : \{0,1\}^{k'} \to \{0,1\}^{kq}$ be a (t_1, ε)-secure pseudorandom generator, and assume that the running time of C is at most t_2. Then $E'_{K'}(X) = C(X \oplus G(K'))$ is $(t_1 - (t_2 + c)q, q, 2\varepsilon)$-LOR secure, for some small constant c.*

Proof. Let E_K denote the encryption scheme when used with truly random K, and $E_{G(K')}$ denote the scheme where the keystream K is generated as the output of the PRG, $K = G(K')$. By Theorem 2,

$$\Pr[A^{E_K \circ \S_1} = 1] = \Pr[A^{E_K \circ \S_0} = 1].$$

Next, we apply the triangle inequality:

$$\begin{aligned}
&|\Pr[A^{E_{G(K')} \circ \S_1} = 1] - \Pr[A^{E_{G(K')} \circ \S_0} = 1]| \\
&\quad \le |\Pr[A^{E_{G(K')} \circ \S_1} = 1] - \Pr[A^{E_K \circ \S_1} = 1] + \Pr[A^{E_K \circ \S_0} = 1] - \Pr[A^{E_{G(K')} \circ \S_0} = 1]| \\
&\quad \le |\Pr[A^{E_{G(K')} \circ \S_1} = 1] - \Pr[A^{E_K \circ \S_1} = 1]| + |\Pr[A^{E_{G(K')} \circ \S_0} = 1] - \Pr[A^{E_K \circ \S_0} = 1]|.
\end{aligned}$$

We will show that both terms on the right-hand side are small.

We can define an adversary B_i that attempts to distinguish between K and $G(K')$ as follows:

$$B_i(z) = A^{E_z \circ \S_i}.$$

B_i can be thought of as a program that mimics the behavior of the attacker A and responds to A's oracle queries by executing the basic cryptosystem with keystream z. If A runs in time $t_1 - t_2 q - cq$, and if C runs in time t_2, then B_i will run in time t_1, since the extra overhead (beyond that of C) for answering each of A's oracle queries is a small constant. Now, because the pseudorandom generator G is assumed to be (t_1, ε)-secure and because B_i runs in time at most t_1, it follows that B_i's advantage at breaking G is minimal:

$$|\Pr[B_i(G(K')) = 1] - \Pr[B_i(K) = 1]| \le \varepsilon.$$

Substituting the definition of B_i into the previous equation, we see that

$$|\Pr[A^{E_{G(K')} \circ \S_i} = 1] - \Pr[A^{E_K \circ \S_i} = 1]| \le \varepsilon.$$

This completes the proof. □

We see that the security of our encryption scheme is directly dependent on the security of the pseudo-random generator. If we believe that we are using a strong PRG, then the stream cipher encryption scheme will also be secure.

5 Related Work

A problem closely related to source coding with side information has been studied in the communication complexity literature (cf., [15, Exercise 4.55, p.64]). Suppose Alice holds a n-bit binary string X, and Bob holds S, chosen so that the Hamming distance between X and S is at most d. How many bits does it take for Alice to communicate X to Bob?

This problem has a well-known solution [16–18] [19, §6] [20, Example 4] [21, Remark 5.1]: Pick any linear error correcting code $\mathcal{E} \subseteq \{0,1\}^n$ that corrects up to d errors; write $X = C \oplus U$, where $C \in \mathcal{E}$ is a codeword and U is a syndrome, and send U to Bob; Bob can apply the decoding algorithm to $U \oplus S = C \oplus (X \oplus S)$ to obtain C, and then Bob can compute $X = C \oplus U$. Also, if X and S are drawn from correlated i.i.d. binary sources, then for all $\epsilon > 0$, the Hamming distance between X and S is at most $n \cdot (\Pr[X \neq S] + \epsilon)$, except with exponentially small probability, so this yields a source coding algorithm with exponentially small decoding error for the special case of i.i.d. binary sources.

As a result, one can build protocols for source coding with side information out of any high-rate linear error-correcting code. The best provable rates for explicit constructions can be found in [10, 22].

6 Conclusions

In this work, we have examined the possibility of first encrypting a data stream and then compressing it, where the compressor does not have knowledge of the encryption key. The encrypted data can be compressed using distributed source coding principles, because the key will be available at the decoder. Our principal contribution is in the observation that the problem of compressing encrypted data is a special case of the source coding with side information problem. We have studied both the compression efficiency and the cryptographic security aspects of this problem. It is an interesting open problem to extend our work to encryption schemes beyond the stream cipher.

7 Acknowledgements

We thank the anonymous reviewers for many extremely helpful comments.

References

1. B. Schneier, *Applied Cryptography.* New York: Wiley, 1996.
2. D. K. Slepian and J. K. Wolf, "Noiseless Coding of Correlated Information Sources," *IEEE Transactions on Information Theory*, vol. 19, pp. 471–480, July 1973.
3. I. Csiszar, "Linear Codes for Sources and Source Networks: Error Exponents, Universal Coding," *IEEE Transactions on Information Theory*, vol. 28, pp. 585–592, July 1982.

4. T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
5. S. S. Pradhan and K. Ramchandran, "Distributed Source Coding Using Syndromes (DISCUS): Design and Construction," in *Proceedings of the Data Compression Conference (DCC)*, (Snowbird, UT), March 1999.
6. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation," in *38th Symposium on Foundations of Computer Science*, (Miami Beach, FL), October 1997.
7. S. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice Hall, 1995.
8. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," in *IEEE International Conference on Communications*, (Geneva, Switzerland), May 1993.
9. R. G. Gallager, *Low Density Parity Check Codes*. PhD thesis, MIT, Cambridge, MA, 1963.
10. M. Sipser and D. A. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1710–1722, November 1996.
11. D. Schonberg, K. Ramchandran, and S. S. Pradhan, "LDPC Codes Can Approach the Slepian Wolf Bound for General Binary Sources," in *40th Annual Allerton Conference*, October 2002.
12. A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of Binary Sources with Side Information Using Low-Density Parity-Check Codes," in *IEEE Communication Letters*, 2002.
13. J. Garcia-Frias and Y. Zhao, "Compression of Correlated Binary Sources Using Turbo Codes," in *IEEE Communication Letters*, October 2001.
14. A. Aaron and B. Girod, "Compression with Side Information Using Turbo Codes," in *IEEE Data Compression Conference*, April 2002.
15. E. Kushilevitz and N. Nisan, *Communication Complexity*. New York: Cambridge University Press, 1997.
16. H. Witsenhausen and A. D. Wyner, "Interframe coder for video signals," 1980. United States Patent Number 4,191,970.
17. A. E. Gamal and A. Orlitsky, "Interactive data compression," in *Proceedings 25th IEEE Symposium on Foundations of Computer Science*, pp. 100–108, October 1984.
18. A. Orlitsky, *Communication Issues in Distributed Computing*. PhD thesis, Stanford University, Electrical Engineering Department, 1986.
19. T. Feder, E. Kushilevitz, M. Naor, and N. Nisan, "Amortized communication complexity," *SIAM Journal on Computing*, vol. 24, no. 4, pp. 736–750, 1995.
20. A. Orlitsky, "Interactive communication of balanced distributions and of correlated files," *SIAM J. Discret. Math.*, vol. 6, pp. 548–564, November 1993.
21. G. Cormode, M. Paterson, S. C. Sahinalp, and U. Vishkin, "Communication complexity of document exchange," in *Proceedings of the 11th annual ACM-SIAM symposium on Discrete algorithms*, pp. 197–206, Society for Industrial and Applied Mathematics, 2000.
22. M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson, "Randomness conductors and constant-degree lossless expanders," in *Proceedings of the 34th annual ACM symposium on Theory of computing*, pp. 659–668, ACM Press, 2002.