ON BLIND COMPRESSION OF ENCRYPTED DATA APPROACHING THE SOURCE ENTROPY RATE

Daniel Schonberg, Stark C. Draper, and Kannan Ramchandran

Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA, USA 94720
phone: + (1) 510-643-4034, email: {dschonbe, sdraper, kannanr}@eecs.berkeley.edu
http://basics.eecs.berkeley.edu

ABSTRACT

Traditional data transmission over an insecure noiseless channel consists of first compressing data for efficiency and then encrypting it for security. Reversing the order of these operations is considered in Johnson et al. [6]. The central realization is that while the raw and encrypted data are statistically independent, the encrypted data and key sequence are not. If distributed source coding techniques are used to compress and to jointly decode and decrypt the data, reversing the order need not lead to performance reductions in either communication efficiency or security. In this paper, we build on this work by considering systems that must operate without knowledge of the underlying source statistics. We present and analyze an incremental scheme based on exponentially increasing block lengths that is designed to balance the resolution rate of parameter estimation with the redundancy rate of communication. We show that the redundancy at best declines proportional to the inverse of the square root of the block length. We implement these ideas using low-density parity check (LDPC) codes. In practical tests to transmit a binary source of $100,000$ bits, ideally compressible to $17,912$ bits with perfect knowledge and an ideal code, required only $26,787$ bits. In comparison, to transmit this source with full knowledge of the source statistics required $21,704$ bits.

1. INTRODUCTION

Existing systems offering efficient and secure communication over an insecure channel first compress the raw data, and then encrypt the compressed source. Security is obtained, e.g., by using a one-time pad, resulting in an encrypted source (cypher-text) that is statistically independent of the raw source. If, however, the source is not compressed before encryption, then the independence of the source and cypher-text might make it seem that data compression is impossible.

However, in [6] it is shown that by making use of distributed source coding techniques, the dependence between the cypher-text and key can be exploited to make lossless compression of the cypher-text possible. Indeed, based on ideas of distributed source coding pioneered by Slepian and Wolf [2] it is demonstrated in [6] that when decoding and decryption are performed jointly, and conditionally on the key sequence, then the cypher-text is as compressible as the original source.

A complication arises because encryption and compression are not performed by the same agent, and therefore the compression system may not know the statistics of the underlying source. Further, since the cypher-text is statistically independent of the source, these statistics cannot be learned on-line by the encoder through observation of the cypher-text. For these reasons, techniques of universal Slepian-Wolf coding that do not require knowledge of the underlying source statistics are particularly relevant. The schemes developed in [6], however, require that the compressor know the entropy rate of the source.

THIS RESEARCH WAS SUPPORTED BY THE NSF UNDER GRANT CCR-0325311.

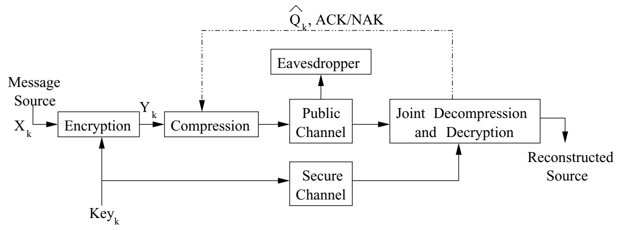

Figure 1: *The source is first encrypted and then compressed. The compressor does not have access to the key used in the encryption step. At the decoder, decompression and decryption are performed jointly.*

In this work we propose an incremental transmission scheme that uses feedback to adapt the Slepian-Wolf rate to the underlying entropy rate. Incremental and universal Slepian-Wolf schemes are not new. Indeed, as is described, e.g., in [8, 3], universal variable-rate Slepian-Wolf techniques exist that can achieve compression to a rate $H(y|k)$ for arbitrary joint source and side information statistics (not just Bernoulli as we focus on here). However, in those papers impractical very-high complexity decoders are used. In this paper we develop a strategy that can be paired with low-complexity graph-based Slepian-Wolf codes.

The feedback required by our approach is minimal, consisting of acknowledgments and summary statistics. We analyze the scheme and show that it achieves a redundancy proportional to the inverse of the square root of the block length. We then describe the results of an implementation of these ideas using low density parity check (LDPC) codes, and give simulation results. A block diagram of this system is give in Figure 1.

This paper is organized as follows. The system model and proposed protocol are described in Section 2. Its performance is analyzed in Section 3. A practical construction using LDPC codes is then described in Section 4 along with simulation results. We conclude the paper in Section 5 with a discussion of future work.

2. SYSTEM MODEL AND PROTOCOL

The source and encryption model are as follows. The source x_i is a sequence of independent identically distributed (i.i.d.) Bernoulli-Q binary random variables. The source is encrypted by adding it to a one-time pad, a binary key sequence k_i that is i.i.d. Bernoulli-0.5. The resulting sequence $y_i = x_i \oplus k_i$ is fed to the encoder, while the key sequence k_i

is made available to the decoder.

As described in [6], by using Slepian-Wolf codes, one need only transmit at a rate equal to $H(y|k) = H(x \oplus k|k) = H(x) = H(Q)$, where $H(Q)$ is the entropy of a Bernoulli-Q source. Thus, even though the encoder observes a perfectly protected source (since k is a one-time pad), if Q is known it can compress the source as much as if it had observed the source unencrypted.

In the current setting we allow the compression system a reverse channel through which estimates of Q can be fed back to the encoder. At time k the encoder maintains a state variable $\hat{Q}_k$, corresponding to its current estimate of the statistics of x_i, and a rate-margin design parameter $\epsilon_k > 0$. We set $\hat{Q}_0 = 0.5$. The system works as follows:

1. The encoder divides the encrypted sequence into block of length l_k, indexed by $k = 1, 2, \ldots$. We use $n_k = \sum_{j=1}^{k} l_j$ to denote the cumulative block-length. The k-th block corresponds to symbols $y_{n_{k-1}+1}^{n_k}$. By choosing the cumulative block-length to double every transmission, i.e., $n_k = 2n_{k-1}$, the error analysis simplifies somewhat. We assume this choice for the duration of the paper.
2. At step k, the encoder encodes $y_{n_{k-1}+1}^{n_k}$ using a rate-R_k Slepian-Wolf code, where R_k depends on the estimate $\hat{Q}_k$ and margin ϵ_k as $R_k(\hat{Q}_{k-1}, \epsilon_k) =$
$$\begin{cases} \max[H(\hat{Q}_{k-1}+\epsilon_k), H(\hat{Q}_{k-1}-\epsilon_k)], \\ \qquad\qquad\qquad \text{if } \epsilon_k < |\hat{Q}_{k-1} - 0.5|, \\ 1, \qquad\qquad\quad \text{else,} \end{cases} \tag{1}$$
i.e., the rate used is $H(\hat{Q}_{k-1})$ plus a margin. The various cases in (1) come into play depending on whether $\hat{Q}_{k-1}$ is greater than or less than one-half, or within the margin ϵ_k of one-half.[1]
3. The decoder attempts to decode. If it can, it sends an acknowledgment to the transmitter, along with the updated estimate $\hat{Q}_k$. The estimate $\hat{Q}_k$ is simply the proportion of 1s (or 0s) observed thus far. The feedback following the kth block can therefore be accomplished with $1 + \log_2 l_k$ bits. If the decoder cannot decode reliably, it sends a "NAK", i.e., a request for retransmission. It holds off sending the updated estimate until it receives the retransmission.[2]
4. If the encoder receives an acknowledgment, it moves onto the next block, using the updated estimate of Q. If the encoder receives a request for retransmission, it sends the sequence $y_{n_{k-1}+1}^{n_k}$ uncompressed.[3]

To help cement the protocol, we provide pseudo-code,

```
• INITIALIZATION
  – Set $\hat{Q}_o = 0.5$. Transmit block 1 of $l_1$ symbols.
  – Receiver feeds back ACK and estimate $\hat{Q}_1$.
• WHILE (Bits available to send)
  – Transmit next $l_k$ symbols at rate $R_k(\hat{Q}_{k-1}, \epsilon_k)$.
  – IF receiver fails to decode
    ∗ Receive sends back NAK
    ∗ Transmitter resends data block uncompressed
  – Receiver ACKs and sends back $\hat{Q}_k$.
```

[1]If we used the more standard choice, i.e., $H(\hat{Q}_{k-1}) + \epsilon_k$, we could avoid the multiplicity of cases of (1). However, by expressing the margin in the current manner, our analysis simplifies, resulting in some nice close-form error expressions. Note that we can choose to add the margin directly to $\hat{Q}$ only because we focus on binary sources.

[2]We assume the code has perfect error-detection capabilities.

[3]More efficient hybrid-ARQ-type retransmission strategies can be used. But, this strategy simplifies the error analysis, and only leads to a marginal loss in efficiency.

If c_k is the transmission rate of block k, we want to minimize the expected transmission rate $E[c_k]$ of the scheme. $E[c_k]$ equals the expected rate $E[R_k]$ of the Slepian-Wolf code plus the probability of a decoding error times $\ln 2$ (1 bit/symbol or $\ln 2$ nats/symbol to account for its uncompressed retransmission). We use e_k to denote the event of a decoding error on block k. Ideally, we minimize the following cost,
$$E[c_k] = \Pr[e_k] \ln 2 + E[R_k]. \tag{2}$$
We choose to express the rate in nats to simplify subsequent notation.

3. ANALYSIS

In this section we show that choosing $\epsilon_k = K(Q)/\sqrt{l_k}$ (for a function $K(\cdot)$) minimizes the cost (2). As one would expect, the form of ϵ_k should be chosen dependent on Q. However, Q is unknown, so in practice we would pick the constant as $K(\hat{Q}_{k-1})$. The dependence on $\sqrt{l_k}$ is also somewhat intuitive. The standard deviation in the best estimate of Q drops as $1/\sqrt{n_{k-1}} = 1/\sqrt{l_k}$.

Without loss of generality, in this analysis we assume that $Q + \epsilon_k < 0.5$. In Section 3.1 we first bound the first term of (2), the probability of a decoding error. In Section 3.2 we bound the second term, the expected rate used during the Slepian-Wolf transmission. In Section 3.3 we put the results together to choose ϵ_k.

3.1 Probability of Decoding Error

We assume that the scheme uses Slepian-Wolf codes that succeed as long as the entropy rate of the kth block is below the transmission rate, i.e., $H(x_{n_{k-1}+1}^{n_k}) < R_k$. We use standard large deviation techniques following the results of [2] to bound the probability of decoding error on the kth block.

$$\Pr[e_k] = \Pr[H(x_{n_{k-1}+1}^{n_k}) > R_k] \tag{3}$$
$$= \sum_P \sum_{x^{n_{k-1}} \in \mathcal{T}_P} p(x^{n_{k-1}}) \Pr[H(x_{n_{k-1}+1}^{n_k}) > R_k] \tag{4}$$
$$= \sum_P \sum_{x^{n_{k-1}} \in \mathcal{T}_P} p(x^{n_{k-1}}) \sum_{\substack{\tilde{P} \text{ s.t.} \\ H(\tilde{P}) > R_k(P, \epsilon_k)}} \sum_{x_{n_{k-1}+1}^{n_k} \in \tilde{P}} p(x_{n_{k-1}+1}^{n_k}). \tag{5}$$

In (3) the rate R_k is random, depending on the empirical distribution of the first n_{k-1} source symbols and the margin ϵ_k via (1). In (4) we use P to denote this empirical distribution. Since $\hat{Q}_{k-1} = P$, plugging P into (1) gives the, now non-random, rate used $R_k(P, \epsilon_k)$. We continue as $\Pr[e_k] \le$

$$\sum_P \sum_{\substack{\tilde{P} \text{ s.t.} \\ H(\tilde{P}) > H(P+\epsilon_k)}} \exp\{-n_{k-1} D(P\|Q) - l_k D(\tilde{P}\|Q)\} \tag{6}$$
$$\le \sum_P \sum_{\tilde{P}} \exp\{-l_k \min_{\substack{P, \tilde{P} \text{ s.t.} \\ H(\tilde{P}) \ge H(P+\epsilon_k)}} [D(P\|Q) + D(\tilde{P}\|Q)]\} \tag{7}$$
$$\le (l_k + 1)^2 \exp\{-l_k [D(P^*\|Q) + D(P^* + \epsilon_k\|Q)]\} \tag{8}$$

In (6) we use $p(x^{n_{k-1}}) = \exp\{-n_{k-1}[H(P) + D(P\|Q)]\}$ for all sequences $x^{n_{k-1}} \in \mathcal{T}_P$, and $|\mathcal{T}_P| \le \exp\{n_{k-1} H(P)\}$, see [2]. In (7) we use $l_k = n_{k-1}$, and the minimization is over all distributions, not just those that are types. In (8) P^* is the minimizing distribution. After minimization the exponent does not depend on P or $\tilde{P}$. We sum over all binary types of length l_k, of which there are $l_k + 1$.

The error exponent of the probability of decoding error depends on the unknown distribution Q. We study this exponent to determine a good choice for ϵ_k. To do this, we solve for P^* assuming a fixed ϵ_k.

$$\begin{aligned}
&\frac{d}{dP}[D(P\|Q) + D(P+\epsilon_k\|Q)] \\
&= \frac{d}{dP}\left[P\ln\frac{P}{Q} + (1-P)\ln\frac{1-P}{1-Q}\right. \\
&\left. +(P+\epsilon_k)\ln\frac{P+\epsilon_k}{Q} + (1-P-\epsilon_k)\ln\frac{1-P-\epsilon_k}{1-Q}\right] \\
&= \ln\left[\frac{P(P+\epsilon_k)(1-Q)^2}{(1-P)(1-P-\epsilon_l)Q^2}\right]. \qquad (9)
\end{aligned}$$

Setting (9) equal to zero, and solving for P using the quadratic equation gives $P^* =$

$$-\frac{\epsilon_k}{2} - \frac{2Q^2 - \sqrt{\epsilon_k^2(1-2Q)^2 + 4Q^2(1-Q)^2}}{2(1-2Q)}. \qquad (10)$$

For any choice of ϵ_k, and any source distribution Q, this value of P^* determines the dominant source of decoding error. Using (10) in (8) yields a bound on the decoding error for this protocol. Note that because $D(P\|Q)$ is convex in its arguments, $P^* \leq Q \leq P^* + \epsilon_k$.

The error exponent $D(P^*\|Q) + D(P^*+\epsilon_k\|Q)$ has a particularly simple form when ϵ_k is small. We define $P^* = Q - \bar{\epsilon}$ and $P^* + \epsilon_k = Q + \bar{\bar{\epsilon}}$, where $\epsilon_k = \bar{\epsilon} + \bar{\bar{\epsilon}}$. By the convexity property just discussed $\bar{\epsilon}, \bar{\bar{\epsilon}} > 0$. With these definitions, we approximate the error exponent when ϵ_k is small.

$$\begin{aligned}
&D(P^*\|Q) + D(P^*+\epsilon_k\|Q) = D(Q-\bar{\epsilon}\|Q) + D(Q+\bar{\bar{\epsilon}}\|Q) \\
&= (Q-\bar{\epsilon})\ln\left[1 - \frac{\bar{\epsilon}}{Q}\right] + (1-Q+\bar{\epsilon})\ln\left[1 + \frac{\bar{\epsilon}}{1-Q}\right] \\
&+ (Q+\bar{\bar{\epsilon}})\ln\left[1 + \frac{\bar{\bar{\epsilon}}}{Q}\right] + (1-Q-\bar{\bar{\epsilon}})\ln\left[1 - \frac{\bar{\bar{\epsilon}}}{1-Q}\right] \\
&\simeq \frac{\bar{\epsilon}^2 + \bar{\bar{\epsilon}}^2}{2Q(1-Q)} + \frac{(\bar{\epsilon}^3 - \bar{\bar{\epsilon}}^3)(1-2Q)}{2Q^2(1-Q)^2} \qquad (11) \\
&\geq \frac{\epsilon_k^2}{4Q(1-Q)} \geq \epsilon_k^2 \qquad (12)
\end{aligned}$$

In (11) we use $\ln[1+x] \simeq x - x^2/2$. Writing $\bar{\epsilon}^2 + \bar{\bar{\epsilon}}^2 = (\bar{\epsilon}+\bar{\bar{\epsilon}})^2 - 2\bar{\epsilon}\bar{\bar{\epsilon}} = \epsilon_k^2 - 2\bar{\epsilon}\bar{\bar{\epsilon}}$, one can see that (11) is minimized under the constraint for small ϵ_k by selecting $\bar{\epsilon} = \bar{\bar{\epsilon}} = \epsilon_k/2$. Choosing $Q = 0.5$ lower-bounds the quadratic approximation of the error exponent.

In Figure 2 we plot the error exponent and quadratic approximation to it for $Q = 0.05$ and $Q = 0.3$. The approximation (12) is quite good, even for large values of ϵ_k. For $Q = 0.3$, one can barely differentiate the quadratic approximation to the full solution. The lowest curve in Figure 2 is the lower-bound to the quadratic approximation with $Q = 0.5$.

3.2 Bounding the expected Slepian Wolf rate, $E[R_k]$

In order to minimize the cost (2) we must also take into account the second term of (2), $E[R_k]$.

$$\begin{aligned}
&E[R_k] \leq \Pr[H(x^{n_{k-1}}) \leq H(Q+\gamma)]H(Q+\gamma+\epsilon_k) \\
&+ \Pr[H(x^{n_{k-1}}) > H(Q+\gamma)]\ln 2 \qquad (13) \\
&\leq H(Q+\gamma+\epsilon_k) + \ln 2 \sum_{\substack{P \text{ s.t.} \\ H(P) > H(Q+\gamma+\epsilon_k)}} \sum_{x^{n_{k-1}} \in \mathcal{T}_P} p(x^{n_{k-1}}) \qquad (14) \\
&\leq H(Q+\gamma+\epsilon_k) + \ln 2 \sum_{\substack{P \text{ s.t.} \\ H(P) > H(Q+\gamma)}} \exp\{-n_{k-1}D(P\|Q)\} \\
&\leq H(Q+\gamma+\epsilon_k) + \ln 2(l_k+1)\exp\{-l_k D(Q+\gamma\|Q)\} \qquad (15)
\end{aligned}$$

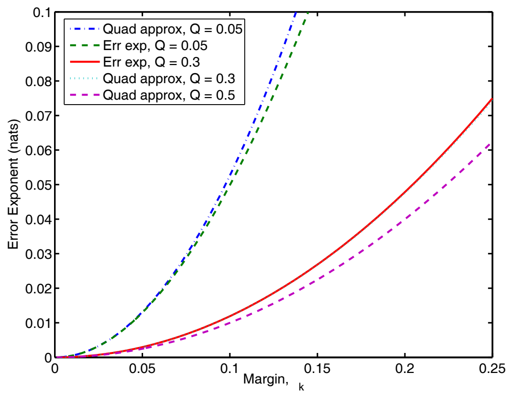

Figure 2: *Error exponents and quadratic approximations for $Q = 0.05$ and $Q = 0.3$.*

In (13) we split the expected rate into two events. The first is a high-probability event that occurs when the realized entropy is below the source entropy plus a margin γ. The second is a low-probability event that occurs when the realized entropy is large. In the former case, the code rate is upper bounded by $H(Q+\gamma+\epsilon_k)$, while in the latter it is upper bounded by $\ln 2$. In (14) we upper bound the probability of the high-probability event by one, and analyze the low-probability event using techniques similar to those used in Section 3.1. As in that section, we also examine the small-γ region which gives,

$$D(Q+\gamma\|Q) \simeq \frac{\gamma^2}{2Q(1-Q)}. \qquad (16)$$

3.3 Optimizing the rate margin, ϵ_k

We can now understand how ϵ_k should be chosen. It is easiest to see this in the small-ϵ_k region. Indeed, the small-ϵ_k region is of great interest as we want to be as efficient as possible for large data files.

Noting that probability of $\Pr[e_k]$ in (3) and $\Pr[H(x^{n_{k-1}}) > H(Q+\gamma)]$ in (13) can both be upper bounded by one, then substituting (8), (10),(15), and (16) into (2) gives

$$\begin{aligned}
&E[c_k] = E[R_k] + \ln 2\Pr[e_k] \leq H(Q+\gamma+\epsilon_k) \\
&\quad + \ln 2 \min\left[1, (l_k+1)\exp\left\{-l_k\frac{\gamma^2}{2Q(1-Q)}\right\}\right] \\
&\quad + \ln 2 \min\left[1, (l_k+1)^2\exp\left\{-l_k\frac{\epsilon_k^2}{4Q(1-Q)}\right\}\right] \qquad (17)
\end{aligned}$$

To give the best bound, we want to pick γ as small as possible. Picking $\gamma = \epsilon_k/\sqrt{2}$ balances the exponents. We combine the exponential terms and use a Taylor series expansion of the entropy around Q, giving:

$$\begin{aligned}
&E[c_k] \leq H(Q) + \frac{1+\sqrt{2}}{\sqrt{2}}\ln\left[\frac{1-Q}{Q}\right]\epsilon_k \\
&\quad + 2\ln 2\min\left[1, (l_k+1)^2\exp\left\{-l_k\frac{\epsilon_k^2}{4Q(1-Q)}\right\}.\right] \qquad (18)
\end{aligned}$$

In (18), the second term is linear in ϵ_k, so we want to pick ϵ_k as small as possible. However, the third term constraints this choice. The margin ϵ_k must go to zero slower than $1/\sqrt{l_k}$, else the polynomial in l_k that pre-multiplies the exponent will dominate. Note that to study the trade-off for small Q, one must use a better approximation to the entropy function than the quadratic one we used. For example $c_k - H(Q)$ should always be less than one bit.

4. PRACTICAL IMPLEMENTATION

In this section, we discuss a practical implementation of the above protocol. Our implementation uses the LDPC based Slepian-Wolf construction of [7]. These ideas can be implemented with any other Slepian-Wolf codes. With more powerful codes, we will see an increase in performance.

LDPC codes [5] are a class of graph-based capacity-approaching linear block codes. They are usually decoded using the sum-product algorithm, an inference algorithm that is exact on trees. Although not exact on "loopy" graphs (such as those that describe LDPC codes), in practice decoding performance is very good. As the algorithm progresses, it either converges to a solution that satisfies the code's constraints (most likely the maximum likelihood solution), or fails to converge at all. We use the latter events to indicate detected decoding errors for our protocol.

LDPC codes have two characteristics that do not match the assumptions of our analysis. First, like almost all good codes, LDPC codes do not respond to all sequences of a particular type equivalently. Thus, these codes do not match our assumption of when a decoding error will occur. Second, LDPC based Slepian-Wolf systems approach the minimal compression rate bound only for large block lengths. As is discussed in [7], extra redundancy is needed for the short block-lengths used for the first few blocks of our protocol.

In our implementation, cypher-text blocks are compressed by finding their syndrome with respect to a LDPC code's parity check matrix. The syndrome is the compressed source information transmitted to the decoder. The decoder runs the sum product algorithm where the syndrome serves as the code constraints and the encryption key for determining likelihood ratios. The likelihoods are determined using the distribution estimate from the previous blocks, $\hat{Q}_{k-1}$.

In each of the simulations, blocks of $100,000$ source symbols are generated according to a Bernoulli-Q distribution. The initial block-length is set to 100, and each successive block length equals the number of bits sent thus far. The LDPC parity check matrices are selected based on the guidelines of [1]. Each matrix corresponds to a particular rate. Since not all rates are available, some additional redundancy is introduced as rates are rounded up to the nearest available rate. The redundancy parameter ϵ_k is set to $\epsilon_k = 2.0/\sqrt{n_{i-1}}$ (arbitrarily, though resulting in good performance).

We plot the results of simulations in Figure 3. In these plots the average cumulative redundancy in percent (averaged over 25 simulations) is plotted versus cumulative block-length, n_k. Cumulative redundancy is defined as $\sum_{j=1}^{k} [c_j - H(Q)]\, l_j / H(Q) n_k$. The results of the system are plotted for 2 different source entropy rates: 0.1791 and 0.1944. As an example, to transmit 10^5 bits for a source entropy $H(Q) = 0.1791$ bits required an average redundancy of 49%, or 8,875 bits more than the 17,912 bit minimum ($26,787$ bits total). In these plots, as n_k grows the overall redundancy declines. In addition, for a source of entropy 0.1791 (the other source is omitted for clarity, but is similar), a bound on the expected cumulative redundancy using (17) is also plotted, as well as the performance assuming full knowledge of the source statistics (based on the results of [7]). Despite the limitations mentioned, and the fact that our bound omits the cost of feedback, our results perform well in relation to our bound.

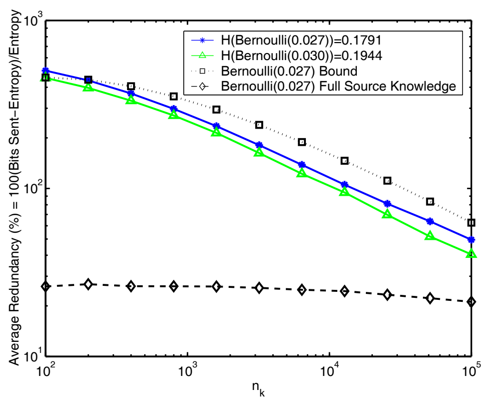

Figure 3: *Averaged results of LDPC based implementation. The horizontal axis is the number of cypher-text bits considered n_k, and the vertical axis is the system redundancy (the percent of bits used above the entropy rate). As the number of bits transmitted grows performance improves. Note, the bound from (17) plotted here excludes feedback costs.*

5. CONCLUSIONS

In this work we have presented a protocol for the blind transmission of an encrypted source using a minimal number of bits. The scheme presented is proven to achieve redundancy proportional to the inverse of the square root of the block length, and requires minimal feedback. We present a practical implementation of this scheme using LDPC codes, taking advantage of their decoding error detection capability.

This work suggests many areas for future work. First, a more efficient retransmission protocol to deal with decoding failures is necessary. In particular, rather than retransmitting the source in the clear, an incremental hybrid-ARQ scheme could provide significant improvement in performance. Other extensions include studying more complex underlying source models. This scheme suggest natural extensions to both sources with memory and higher order alphabets. Still another area for future work is in the consideration of different codes or different decoders for the practical implementation discussed here. The linear programming LDPC Decoder [4] guarantee of maximum likelihood decoding or error detection offers promise.

REFERENCES

[1] A. Amraoui and R. Urbanke. *LdpcOpt*.

[2] I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, New York, 1981.

[3] S. C. Draper. Universal incremental Slepian-Wolf coding. In *42nd Allerton Conference*, October 2004.

[4] J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode linear codes. In *IEEE Trans. on Information Theory*, March 2005. to appear.

[5] R. G. Gallager. *Low Density Parity Check Codes*. PhD thesis, MIT, Cambridge, MA, 1963.

[6] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran. On compressing encrypted data. In *IEEE Trans. on Signal Processing*, volume 52, pages 2992–3006, October 2004.

[7] 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*, pages 576–585, October 2002.

[8] N. Shulman and M. Feder. Source broadcasting with an unknown amount of receiver side information. In *Proc. 2002 Inform. Theory Workshop*, pages 127–130, Bangalore, India, October 2002.