International Symposium on Information Theory and its Applications, ISITA2004
Parma, Italy, October 10–13, 2004

Side-Information Coding with Turbo Codes and its Application to Quantum Key Distribution

Kim-Chi NGUYEN[†], Gilles VAN ASSCHE[†] and Nicolas J. CERF[†]

[†] Centre for Quantum Information and Communication, Ecole Polytechnique
Université Libre de Bruxelles
CP 165, 1050 Brussels, Belgium
E-mail: kimchng@ulb.ac.be, gvanassc@ulb.ac.be, ncerf@ulb.ac.be

Abstract

Turbo coding is a powerful class of error correcting codes, which can achieve performances close to the Shannon limit. The turbo principle can be applied to the problem of side-information source coding, and we investigate here its application to the reconciliation problem occuring in a continuous-variable quantum key distribution protocol.

1. INTRODUCTION

1.1. Side-Information Source Coding

Given a source of two correlated random variables X and Y, the minimal achievable rate of encoding of X is $H(X|Y)$ when Y is given losslessly to the decoder. Surprisingly, this rate is also achievable when Y is known only to the decoder, but not to the encoder [1]. In this setting, Y is called the *side information* and the encoding of X is known as *side-information source coding.*

The construction of efficient side-information source coding schemes is a difficult problem [2]. Recently, turbo codes have shown to be good candidates for this coding application [3].

1.2. The Turbo Principle

Turbo coding was first introduced in 1993 by Berrou et al. [4]. Since then it has been intensively studied and has proved to approach the Shannon limit closer than any other known forward error correcting code. The efficiency of the turbo codes is due to the use of an iterative process at the decoder side and the presence of an interleaver at the encoder side, which adds randomness-like effects to the code.

This work was supported by the Communauté Française de Belgique under grant ARC 00/05-251 and by the IUAP programme of the Belgian government under grant V-18.

Our motivation for studying side-information coding in general, and turbo codes specifically, is described next.

1.3. Quantum Key Distribution

Quantum key distribution (QKD), also called quantum cryptography, allows two parties, Alice and Bob, to share a secret key that can be used for encrypting messages using a classical cipher, e.g., the one-time pad. The main interest of such a key distribution scheme is that any eavesdropping is, in principle, detectable as the laws of quantum mechanics imply that measuring a quantum state generally disturbs it.

To share a secret key, a few steps must be performed. First, quantum states are sent from Alice to Bob, or vice-versa, on the so-called quantum channel. This process gives the two parties correlated random variables, X_A and X_B. Then, using a classical public authenticated channel, Alice and Bob compare a sample of the transmitted data, from which they can determine an upper bound on the amount of information a possible eavesdropper may have acquired. Finally, they distill a common secret key K, which is conventionally a function of X_A.

Secret key distillation [5] usually involves two steps. In the first step, called *reconciliation*, Alice and Bob exchange information over the public authenticated channel in such a way that Bob can recover X_A knowing X_B. The exchanged information is considered known to an eavesdropper. The second step consists in applying a *privacy amplification* protocol [6] to wipe out the enemy's information on both quantum and classical transmissions, at the cost of a reduction in the key length. This reduction is roughly equal to the number of bits known to an eavesdropper [6].

It thus appears clearly that reconciliation should not give more information than necessary on X_A, otherwise resulting in a penalty in the key length. Hence, the interest of investigating the use of (efficient) turbo coding in this context.

1.4. Problems with Interactive Reconciliation

An additional motivation for using turbo codes in the scope of QKD lies in that reconciliation is traditionally performed using interactive protocols, such as Cascade [7]. While they are perfectly suited to discrete QKD protocols, such as BB84 [8], they suffer from both practical and fundamental problems when used for continuous-variable QKD protocols [9]. For a given number of reconciliation bits transmitted from Alice to Bob, interactive protocols impose an additional penalty on the key length over one-way protocols, due to the information leaked from the reconciliation bits originating from Bob. Furthermore, the evaluation of this leaked information depends on the particular eavesdropping strategy, which rules out the use of this method when no assumption on the enemy's side may be made. More details are given in Sec. 3.

Replacing interactive reconciliation protocols by efficient side-information coding is thus another strong motivation for studying this application of turbo coding. Note that the use of other error-correcting codes for secret key distillation was studied by several authors (e.g., see [10, 11]).

2. TURBO CODING WITH SIDE INFORMATION

2.1. Turbo Encoder and Turbo Decoder

A turbo encoder is a parallel concatenation of two, or more, constituent codes separated by one, or more, interleavers. The constituent codes are usually two identical recursive systematic convolutional codes. The input sequence to be encoded is divided into blocks of length N. Each block is encoded by the first encoder and interleaved before passing through the second encoder. In channel coding, the systematic output of the first encoder, along with the parity check bits of both encoders are transmitted through the channel. Such a scheme usually uses rate half constituent encoders, so the overall rate is one third. This rate can be increased by puncturing a fraction of the parity bits.

The turbo decoder consists of two, or more, Soft-In Soft-Out (SISO) maximum likelihood decoders. Those decoders operate in parallel, passing extrinsic information to one another in an iterative way. The error rate is lowered after each iteration but the gain in bit error rate decreases as the number of iterations increases, so for complexity reasons the decoder typically performs between 6 and 20 iterations.

2.2. Decoding Algorithm

Two families of decoding algorithms are commonly used in turbo decoding: Soft Output Viterbi Algorithms (SOVA) and Maximum A Posteriori (MAP) algorithms. The MAP algorithm is more efficient but more complex than the SOVA, but simplified versions of this algorithm, like MAX-Log-MAP and Log-MAP, perform almost as well with a reduced complexity.

Each encoder can be represented by a trellis diagram in which each node represents a state of the shift registers and each branch represents a transition between two states, depending on the value of the input bit. The decoding algorithm must find the path through the trellis corresponding to the correct input sequence. We will here describe the MAP algorithm which was proposed by Bahl et al. [12] in 1974.

Each component decoder takes as input the systematic values from the channel, the parity bits transmitted from the associated component encoder, and the information from the other decoder about the likely values of the bits, which is referred to as *a-priori* information. The decoder then provides not only the estimated bit sequence but also the probability for each bit that it has been decoded correctly. This probability is the *a-posteriori Log Likelihood Ratio* (LLR) and is computed for each bit u_k in the sequence.

$$L(u_k|\overline{y}) = \ln\left(\frac{P(u_k = +1|\overline{y})}{P(u_k = -1|\overline{y})}\right)$$

where $\overline{y}$ is the received sequence, the value +1 represents the transmitted bit 0 and -1 represents the transmitted bit 1. This LLR can be rewritten as:

$$L(u_k|\overline{y}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} P(S_{k-1} = s' \wedge S_k = s \wedge \overline{y})}{\sum_{(s',s)\Rightarrow u_k=-1} P(S_{k-1} = s' \wedge S_k = s \wedge \overline{y})}\right)$$

where $\sum_{(s',s)\Rightarrow u_k=+1}$ sums over the set of transitions that can occur if the input bit $u_k = +1$, and similarly for $\sum_{(s',s)\Rightarrow u_k=-1}$. The probability $P(S_{k-1} = s' \wedge S_k = s \wedge \overline{y})$ can be split into 3 terms:

$$P(S_{k-1} = s' \wedge S_k = s \wedge \overline{y}) = \alpha_{k-1}(s')\,\gamma_k(s',s)\,\beta_k(s)$$

These 3 terms are successively computed by the algorithm.

- $\gamma_k(s', s)$ is the probability that given the trellis was in state s' at time $k-1$, it moves to state s and the received channel sequence for this transition is $\overline{y}_k$.

  $$\gamma_k(s',s) = P(\overline{y}_k|\overline{x}_k)\,P(u_k),$$

  where $P(u_k)$ is the a-priori probability given by the previous component decoder and $P(\overline{y}_k|\overline{x}_k)$ is the probability that given the codeword $\overline{x}_k$ associated with the transition was transmitted the channel sequence $\overline{y}_k$ was received.

- $\alpha_{k-1}(s')$ is the probability that the trellis is in state s' at time $k-1$ and the received channel sequence up to this point is $\overline{y}_{j<k}$. The $\alpha_k(s)$ are computed for each k and each s in a forward recursive manner:

$$\alpha_k(s) = \sum_{\text{all } s'} \alpha_{k-1}(s')\,\gamma_k(s', s),$$

with the initial conditions depending on the initial state of the trellis.

- $\beta_k(s)$ is the probability that given the trellis is in state s at time k the future received channel sequence will be $\overline{y}_{j>k}$. The $\beta_k(s)$ are computed for each k and each s in a backward recursive manner:

$$\beta_{k-1}(s') = \sum_{\text{all } s} \beta_k(s)\,\gamma(s', s),$$

with the initial conditions depending on the final state of the trellis.

Finally the LLR is computed for each bit in the sequence,

$$L(u_k|\overline{y}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_k=+1} \alpha_{k-1}(s')\,\gamma(s',s)\,\beta_k(s)}{\sum_{(s',s)\Rightarrow u_k=-1} \alpha_{k-1}(s')\,\gamma(s',s)\,\beta_k(s)}\right),$$

and this information is passed to the second constituent decoder as a-priori information.

After the last iteration is completed, a final hard decision is taken for each bit, following:

$$u_k = \begin{cases} +1 & \text{if } L(u_k|\overline{y}) \geq 0 \\ -1 & \text{if } L(u_k|\overline{y}) < 0 \end{cases}$$

2.3. Application to Side-Information Source Coding

In the turbo coding principle, the systematic output of the first component encoder is sent through the channel together with the parity bits from the two encoders. Turbo coding can be used for side-information source coding if we consider the input bit sequence as the random variable X, the systematic output of the channel as the side information Y, and the parity bits from the two encoders as the information provided to recover X from Y.

Thus, in practice, Y is a noisy version of X that is known by the receiver, X is encoded with a turbo encoder by the emitter but only the parity bits are transmitted, and the receiver uses those parity bits and Y to recover X by turbo decoding. To achieve a transmission rate close to the Slepian-Wolf limit, an appropriate puncturing pattern must be used to transmit only a fraction of the produced parity bits.

3. RECONCILIATION FOR CONTINUOUS-VARIABLE QKD

Gaussian-modulated QKD protocols using coherent states have shown to deliver higher secret bit rates than those based on single photons while using standard telecom optical components [9]. Since they produce continuous variables (i.e., X_A and X_B are correlated Gaussians), a reconciliation procedure adapted to this situation must be used. We here assume that the variable X_A is converted into bits, as described in [13] and implemented in [9].

Without going into the details, each instance of X_A is transformed into m bits, making m l-bits strings $\{S_i\}_{i\in\{1\ldots m\}}$, each called a *slice*, when a run of the QKD protocol produces l instances of Gaussian variables. These ml bits will serve as input to the privacy amplification protocol. On his side, Bob needs to determine Alice's bit values. For this, he calculates his best estimate of S_i given the l values of X_B, thus producing the l-bit string $\tilde{S}_i$ for each i. ([1]) Using a binary reconciliation protocol, Bob then recovers S_i given his knowledge of $\tilde{S}_i$.

Even though we started from continuous variables, we thus reach a situation where Alice and Bob need to reconciliate the binary string S_i, given that Bob knows the correlated binary string $\tilde{S}_i$. The two strings are related by the error rate e_i, that is the probability that a bit of S_i is not equal to the corresponding bit in $\tilde{S}_i$. Overall, the reconciliation produces $H(S_{1\ldots m})$ uniform bits by disclosing $\sum_{i=1\ldots m} f(e_i)$ bits, with $f(e)$ the number of bits needed to encode a l-bit string given that the decoder knows a correlated string with bit error rate e. The net result is thus $H(S_{1\ldots m}) - \sum_{i=1\ldots m} f(e_i)$.

3.1. Binary Reconciliation

Let us discuss the different options for the binary reconciliation of a given slice with error rate e. Of course, it is always possible to encode S using l bits, so that $f(e) \leq l$.

Using a interactive reconciliation protocol such as Cascade [7] implies that Alice and Bob exchange parities of various subsets of their strings. After running Cascade, Alice and Bob have disclosed RS and $R\tilde{S}$ for

[1]Actually, the slices are corrected sequentially, for $i = 1\ldots m$, so that the estimation of $\tilde{S}_i$ can also depend on the knowledge acquired from the previous corrected slices S_j, $j < i$ [13].

some binary matrix R of size $d \times l$. They thus have communicated the parities calculated over d identical subsets of bit positions. The matrix R and the number d of disclosed parities are not known beforehand but are the result of the interactive protocol, depending on the diverging parities encountered. For Cascade, $d \approx l(1+\xi)h(e)$, where $h(e) = -e \log e - (1-e) \log(1-e)$ and ξ is some small overhead factor $\xi \ll 1$.

In the case of balanced bit strings (i.e., the probabilities of 0 and 1 are the same), the parities RS give Eve d bits of information on S, but $R\tilde{S}$ does not give any extra information since it is merely a noisy version of RS, or stated otherwise, $S \to RS \to R\tilde{S}$ is a Markov chain.

However, in the more general case where we need to take into account that Eve gathered in E some information on both S and $\tilde{S}$ by eavesdropping on the quantum channel, $S|E \to RS|E \to R\tilde{S}|E$ does not necessarily form a Markov chain. Instead, the actual number of bits disclosed during reconciliation, namely $I(RS, R\tilde{S}; S|E)$, must be explicitly evaluated. This quantity is in general larger than d, therefore adding an extra cost due to interactivity.

Furthermore, it is unfortunately impossible to evaluate this quantity without making an assumption on the eavesdropping strategy, since we need to explicitly express the variable E. In [9], this quantity was calculated for the most general assumption within the scope of that paper (i.e., assuming any individual Gaussian eavesdropping strategy). However, beyond this assumption, the calculation loses its validity.

To remove any assumption, a possibility is to upper bound the number of disclosed bits as if both parties disclosed independent information, that is, $I(RS, R\tilde{S}; S|E) \leq 2d \approx 2l(1+\xi)h(e)$. This is unfortunately too expensive in practice, except when e is small, and causes the secret key rate to vanish if this worst-case measure is taken for all slices. (We will use this solution for small e in our application, see Sec. 4.3.)

Another option for reconciliation is of course to use side-information source coding, as we will do in Sec. 4. This provides a non-interactive reconciliation protocol that has the advantages of being independent of the eavesdropping strategy and free of interactivity cost.

4. APPLICATION TO AN EFFICIENT QKD PROTOCOL

4.1. Settings

We used a turbo code as a binary reconciliation protocol in the continuous-variable QKD protocol described in [9].

The component encoders are two 16-state duo-binary recursive systematic convolutional encoders with generator polynomials (23, 35) [14]. The interleaver is a variation of the odd/even interleaver presented by Barbulescu [15] and is detailed in section 4.2. The puncturing pattern depends on the estimated error rate and is chosen to minimize the number of bits sent to Bob to achieve a final BER $< 10^{-6}$.

The decoding algorithm is the Log-MAP algorithm, which is similar to the MAP algorithm but operates in the log-domain. We applied a scheme proposed by Fujii et al. [16], which consists of weighting the extrinsic information exchanged by the two decoders by a factor depending on whether or not the corresponding parity bit has been received for this bit. We performed 18 iterations with block size $N = 10000$.

4.2. Interleaver

Barbulescu [15] introduced an interleaver called *odd/even* designed for rate $\frac{1}{2}$ turbo codes. In rate $\frac{1}{2}$ turbo codes, half the parity bits from each encoder are punctured, so the odd/even interleaver interleaves the bits in such a way that each information bit has its corresponding parity bit transmitted by one and only one encoder. Thus the coding power is uniformly distributed over all the information bits.

For our application, the puncturing pattern depends on the estimated error rate and is different in each case. The interleaver is designed depending on the puncturing pattern in the following way:

1. The information bits are split into 2 groups V_1 and V_2: V_1 contains bits whose parity bit has been transmitted by the first encoder and V_2 contains bits whose parity bit has not been transmitted by the first encoder.
2. The 2 groups are interleaved separately and pseudo-randomly.
3. The bits are reordered in a vector W in the following way: the places in W for which the parity bits will be transmitted by the second encoder are filled with bits from V_2 first and with bits from V_1 if needed. The remaining places in W are then filled with the remaining bits.
4. The vector W contains the reordered bits to be passed to the second encoder.

This procedure prevents us from transmitting the two parity bits from one bit, while another bit has none of his parity bits transmitted. Note that if more than half the parity bits are punctured, some information bits will not have any of their parity bits transmitted, but we are assured that none will have both parity bits transmitted. On the other hand, if less than half the parity bits are punctured, some bits will have both parity bits transmitted, but all bits will have at least one parity bit transmitted.

4.3. Results

Figure 1 shows the number of bits disclosed as a function of the bit error rate. According to these results, we chose the best strategy among three possible options, depending on the bit error rate, so as to minimize the number of disclosed bits. Each binary string S_i was reconciliated using one of these three options, depending on the estimated error rate e_i:

- if $e_i > 15\%$, the string was completely revealed, disclosing l bits of information;
- if $e_i < 0.8\%$, an interactive error correction protocol (Cascade) was preferred and the number of disclosed bits was counted independently for Alice and for Bob;
- otherwise, the turbo coding scheme described above was used.

In Table 1, our results are compared with those of [9] based on the use of reverse reconciliation with estimate of the interactivity cost under assumptions. An example of the processing of each slice is given in Table 2: each slice is reconciliated using full disclosure, Cascade, or a turbo code. For higher losses, the gain on the interactivity cost more than compensates for the higher number of parity bits revealed by a turbo code.

Modulation Variance	Losses (dB)	Results from [9] rate (kbs^{-1})	Cascade and Turbo Code rate (kbs^{-1})
41.7	0	1690	1605
38.6	1.0	470	450
32.3	1.7	185	209
27	3.1	75	81

Table 1: *Net secret key rate with modulation frequency of 800 kHz.*

5. CONCLUSIONS

We have shown that decoupling reconciliation and eavesdropping analysis in continuous-variable QKD protocols by using turbo codes allows close, if not better, results than by using Cascade and an evaluation of interactivity costs under assumptions. Furthermore, this opens the way to enhancing the secret key rate for lossy (long-distance) transmissions, for which the interactivity seems to play a critical role.

Slice	Estimated BER e_i (%)	Binary Correction Protocol	Disclosed Bits	Shannon Limit $h(e_i)$
1	49.68	Full Disclosure	l	0.99
2	34.89	Full Disclosure	l	0.93
3	6.38	Turbo Code	$0.46l$	0.34
4	0.02	Cascade	$2 \times 0.005l$	0.0027
5	6×10^{-12}	Cascade	$2 \times 0.004l$	3×10^{-10}

Table 2: *Disclosed bits for each slice, corresponding to the 2nd row of Table 1.*

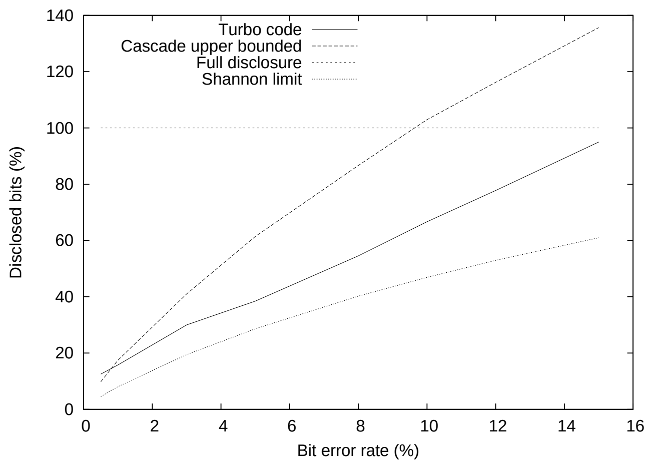

Figure 1: Number of disclosed bits as a function of the bit error rate.

Acknowledgments

We thank Jean Cardinal for suggesting this research topic and for many fruitful discussions.

References

[1] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. 19, no. 4, pp. 471–480, July 1973.

[2] Q. Zhao and M. Effros, "Low complexity code design for lossless and near-lossless side information source codes," in *Proc. IEEE Data Compression Conference*, 2003.

[3] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. IEEE Data Compression Conference*, 2002.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correction coding and decoding: Turbo-codes," *Proc. Int. Conf. Communications*, pp. 1064–1070, May 1993.

[5] U. M. Maurer, "Secret key agreement by public discussion from common information," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 733–742, May 1993.

[6] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, "Generalized privacy amplification," *IEEE Trans. Inform. Theory*, vol. 41, no. 6, pp. 1915–1923, November 1995.

[7] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in *Advances in Cryptology – Eurocrypt'93*, ser. Lecture Notes in Computer Science, T. Helleseth, Ed. New York: Springer-Verlag, 1993, pp. 411–423.

[8] C. H. Bennett and G. Brassard, "Public-key distribution and coin tossing," in *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India.* New York: IEEE, 1984, pp. 175–179.

[9] F. Grosshans, G. Van Assche, J. Wenger, R. Brouri, N. J. Cerf, and P. Grangier, "Quantum key distribution using gaussian-modulated coherent states," *Nature*, vol. 421, pp. 238–241, 2003.

[10] E. Furukawa and K. Yamazaki, "Application of existing perfect code to secret key reconciliation," in *Int. Symp. on Commun. and Inform. Tech.*, 2001, pp. 397–400.

[11] J. Muramatsu, "Secret key agreement from correlated source outputs using ldpc matrices," in *Int. Symp. on Inform. Th.*, 2004, p. 15.

[12] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[13] G. Van Assche, J. Cardinal, and N. J. Cerf, "Reconciliation of a quantum-distributed gaussian key," *IEEE Trans. Inform. Theory*, vol. 50, p. 394, 2004.

[14] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Communications magazine*, vol. 41, pp. 110–116, 2003.

[15] A. Barbulescu, "Iterative decoding of turbo codes and other concatenated codes," Ph.D. dissertation, Uni. of South Australia, 1996.

[16] H. Fujii, T. Saba, and I. Sasase, "Novel decoding algorithm with weighting of extrinsic information for punctured turbo-codes," in *4th European Personal Mobile Communications Conference (EPMCC 2001)*, 2001.