Error Control Watermarking

Deletion-Insertion Correcting Codes and Robust Watermarking

Talk by (Hans) Georg Schaathun
Hard Work by Johann Briffa

Department of Computing
University of Surrey

25 February 2008

UNIVERSITY OF SURREY

UNIVERSITY OF SURREY

Robust Watermarking

Definition

According to Kalker [1]:

> *Robust watermarking is a mechanism to create a communication channel that is multiplexed into original content (the host data). It is required that, firstly, the perceptual degradation of the marked content (host data multiplexed with the auxiliary data) with respect to the original content is minimal and, secondly, that the capacity of the watermark channel degrades as a smooth function of the degradation of the marked content.*

UNIVERSITY OF SURREY

Applications

- Copyright protection
  - Information about copyright holder, licencee, or licence
  - Threat: malicious users
- Authentication and self-recovery
  - Semi-fragile watermarks
    - destroyed by illegitimate changes (doctoring)
    - survives noise and legitimate processes
  - Threat: errors from signal processing

UNIVERSITY OF SURREY

Signal Transformation Attacks

- Additive noise
  1. White noise
  2. Digital compression
  - Well-known effect
  - Quantified by PRNG, Euclidean distance, etc.
- Geometrical distortion
  - Easily causes loss of synchronization
  - Perceptual degradation unquantified
  - Less theory on appropriate error-control codes

Quantum Index Modulation

- The image $\sim$ matrix of pixels
  - Grayscale pixels in $\{0, 1, 2, \dots, 255\}$
  - Colour, e.g. RGB vectors (r, g, b)
- Small errors are not perceptible
- Divide $\{0, 1, \dots, 255\}$ into segments
- Alternate segments encode different bit values
  - 0: $\{0,1,2,6,7,8,12,13,14,18,19,20,24,25,26\dots\}$
  - 1: $\{3,4,5,9,10,11,15,16,17,21,22,23\dots\}$
- The modulator modifies each pixel to encode the transmitted value
  - while minimising the distortion

Example

- Embedded ≪Hello World!≫ using QIM and repetition code

Host

Spatial domain

Transform domain

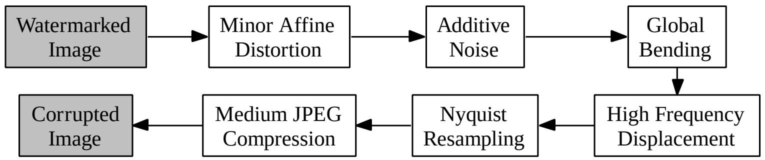
- This is not robust to geometrical attack
  - requires synchronisation

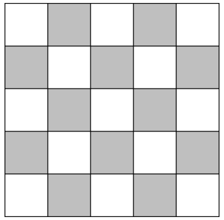
Classes of Distortion

Classification by attack properties:

- Affine attacks
  - Straight lines remain straight
  - Parallel lines remain parallel
  - *Example:* Resolution change; Rotation
  - *Challenge:* Determining the translation matrix
- Cropping / Padding
  - Only part of the original image remains
  - Only part of the result is from the original
  - *Example:* Video aspect conversion
  - *Challenge:* Redundancy & Determining the offset
- Other
  - Nonlinear distortion
  - *Example:* Print & Scan

Stirmark Attack

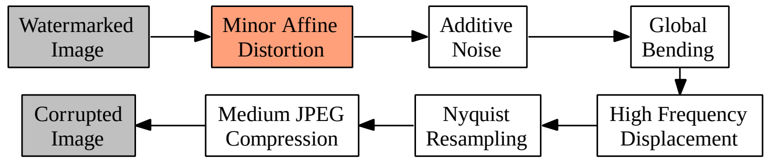


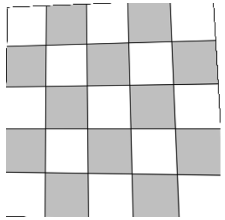


Note:

- Stirmark is a multistep process
- We start with the watermarked image
- To demonstrate the effect, consider the grid shown

Stirmark Attack – Step 1



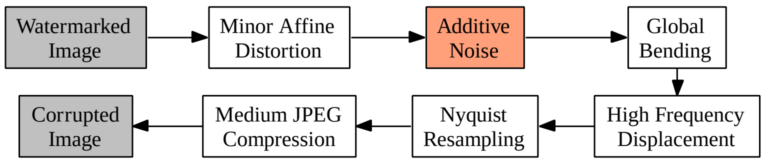


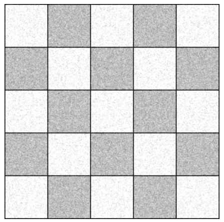
'Affine' distortion is applied as:

- A random shift to the corners
- The rest of the image is linearly stretched accordingly

This is really a *projective* transform

Stirmark Attack – Step 2



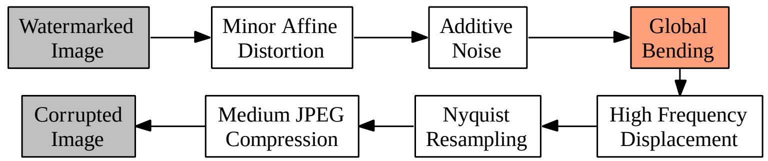


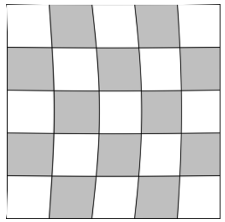
Noise is applied as:

- The addition of a random $\eta(x, y)$ at (x, y)

We assume here the random noise is Gaussian.

Stirmark Attack – Step 3



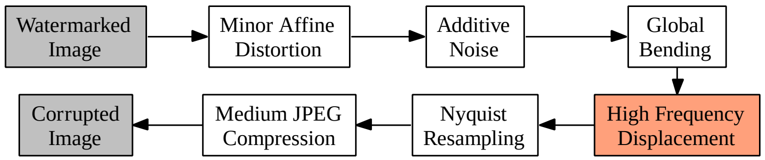


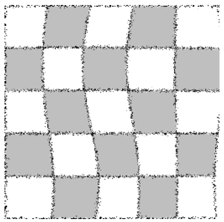
'Bending' is applied as:

- A shift is defined for the center of the image
- The shift reduces to zero at the edges

We assume here that the envelope is sinusoidal.

Stirmark Attack – Step 4





High-frequency distortion is really two components:

- A shift of $\lambda \sin(\omega_x x) \sin(\omega_y y)$ at (x, y)
- A random shift of $n(x, y)$ at (x, y)

We assume here a Gaussian random shift.

Countermeasures

Determining the transformation parameters:

- By comparison with the original
  - Impossible in blind watermarking schemes
- Using a pilot sequence
  - Opens up attacks based on the pilot
- Embedding in an invariant domain
  - No domain is invariant to *all* attacks
- Using error-control coding
  - Channel with Deletions and Insertions
  - Some codes exist – for 1D ...

Existing Codes

Deletion-Insertion Correcting Codes

60-s Levenshtein and others – small codes, correcting one error

1999 Schulman & Zuckerman [2]

2001 Davey & MacKay [3] – *Watermark Codes*

2003 Ratzer [4]

Existing DICC are not ideally suited for application in watermarking:

- Channel model is one-dimensional
- Insertion/Deletion model is discrete
- Error statistics may not correspond to possible attacks

Substitution versus 'Edit' Errors

- Most error-correcting codes designed for *substitution* errors
  - The metric is the Hamming distance
- Deletion-Insertion Correcting codes consider three types of error
  - Insertion, Deletion, and Substitution
  - Insertion and Deletion causes desynchronisation
  - The metric is the Levenshtein distance (or edit distance)

Block Diagram

- Inner 'Watermark' Code – pilot sequence for synchronization
- Outer LDPC Code – corrects resultant substitution errors

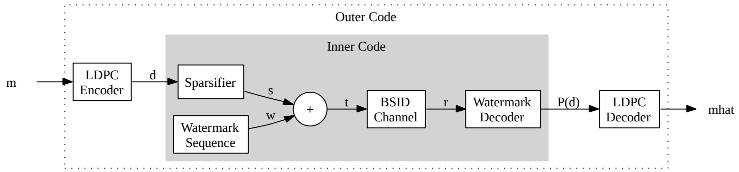


Figure: Structure of a Watermark Code

Watermark Codes

The idea

- Watermark Code $\sim$ Pilot Sequence
  - used for synchronisation
  - random word $\vec{w}$
- Modulation by substitution errors in the pilot sequence
  - encode as a sparse vector $\vec{s}$
  - transmit $\vec{c} = \vec{w} + \vec{s}$

Example – The Sparsifier

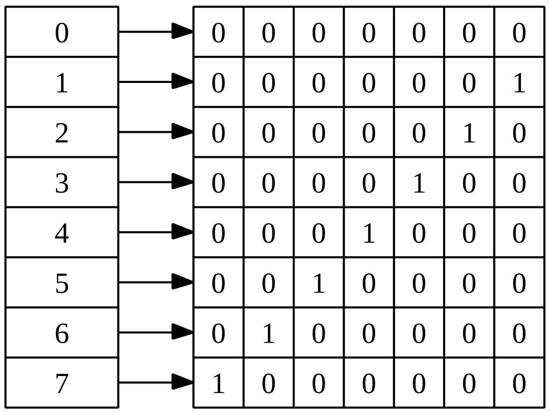


Figure: Sparsifier Mapping

In q-ary (in this case 8-ary)

Out binary (here in groups of 7 bits)

Example – Encoding & Tranmission

Transmit symbol sequence $0, 1, 2, 3, 4$, where $P_s = P_d = P_i = 3\%$

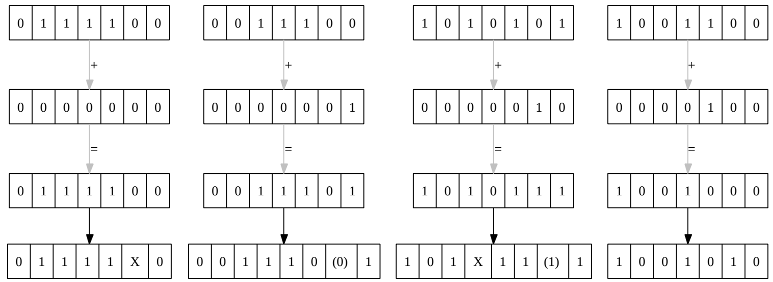


Figure: Encoding & Transmission

Example – Decoder

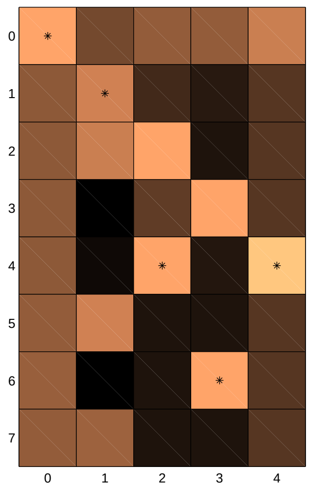


Figure: Likelihood Table

1. For each symbol 'slot', consider:
   - All possible drifts
   - All possible channel edits
   - All possible sparse symbols
2. Compute probability for every sparse symbol
3. Decoder cannot distinguish between:
   - Channel errors (edits)
   - Sparse symbol uncertainty
4. Consider symbol at slot 2...

Example – Decoder

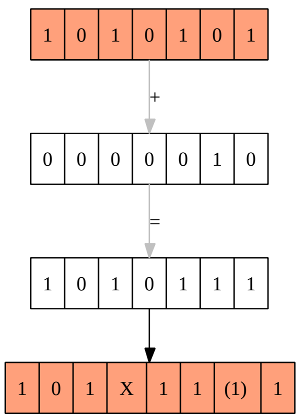


Figure: Slot 2

- Consider only this input drift
- Receiver sees only the 1's and 0's
- Compare received with watermark...
- Possible explanations:

Sym	Edits
2	S_4
4	S_6
4	D_6 (next bit is 1)
	etc.

- Sum probabilities for each symbol
- Symbol 4 seems most likely...

Remarks

Points worth noting about Watermark codes:

- Decoding is very expensive; involves all combinations of:
  - drifts
  - channel edits
  - sparse symbols
- Ambiguity between sparse symbols and channel edits
- Strong error correction is needed

Many Watermarking Applications are not like Real Time Communications

- a full day to process a single image may be acceptable

Davey's Codes

Observation

Watermark codes effectively translate insertions and deletions to substitution errors.

Experiment

We seek to determine the relationship by simulating the Watermark codes used by Davey without additional protection.

WM	k	n	Rate	
a	4	5	0.8	$\frac{4}{5}$
b	4	6	0.6667	$\frac{2}{3}$
c	3	6	0.5	$\frac{1}{2}$
d	3	7	0.4286	$\frac{3}{7}$

Table: Watermark codes used by Davey

Davey's Codes

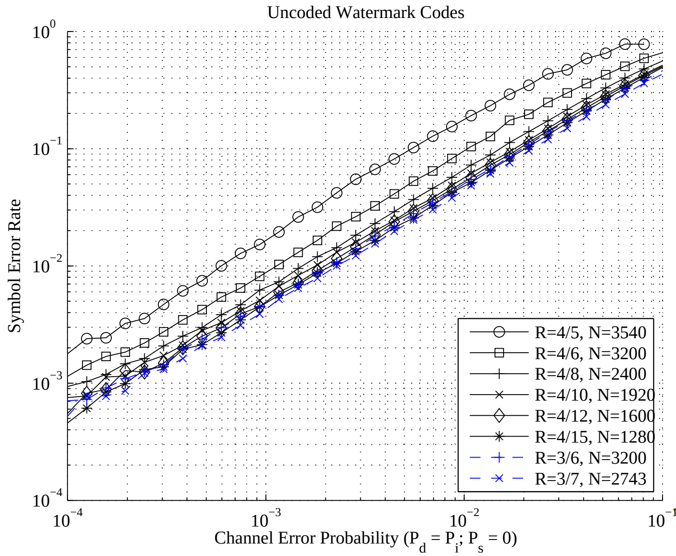


Figure: SER performance of unprotected Watermark codes

Davey's Codes

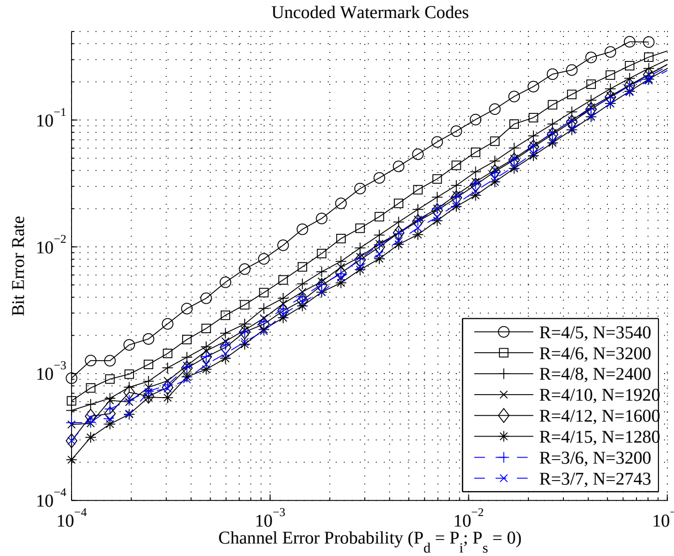


Figure: BER performance of unprotected Watermark codes

Davey's Codes

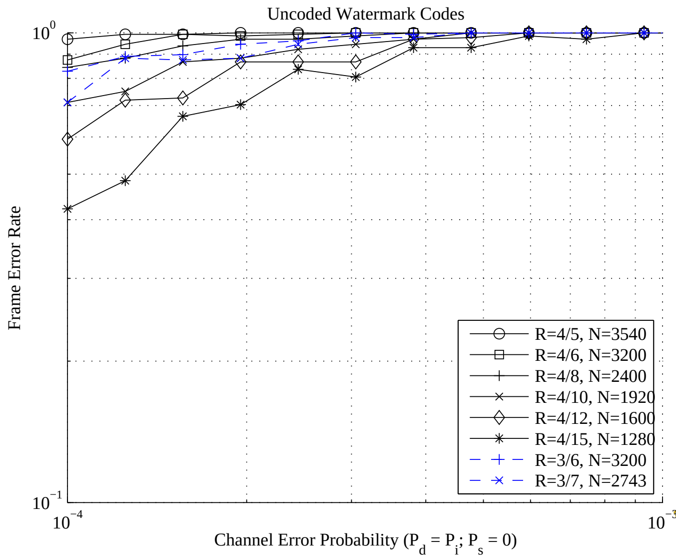


Figure: FER performance of unprotected Watermark codes

Lesson learnt

- The Davey inner codes effectively translate deletion/insertion errors to substitution errors
- A rate of 3/7 appears to be optimal, and 3/6 is close.
- Any standard error-correcting code can solve the remaining problem
- Davey used LDPC outer codes
- We wanted to try turbo codes instead
  - to take advantage of our basis of knowledge and software
  - ... and maybe they work better?

Introduction

Turbo codes are:

- (Large) block codes
- Built by concatenating convolutional codes
  - Usually parallel concatenated
  - Often two codes only
  - Normally the same code is repeated
  - Must use recursive codes
- Decoded iteratively
  - Must be soft-decision decoder
  - Original decoder is BCJR (optimizes BER)
  - Later SOVA was introduced (optimizes FER)
- Capable of operating close to Shannon limit

Block Diagram – Encoder

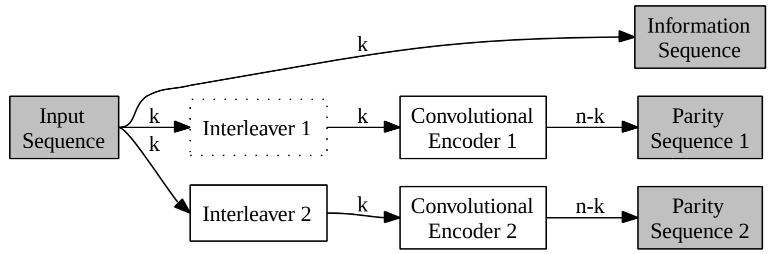


Figure: Structure of a Turbo Encoder

Block Diagram – Encoder

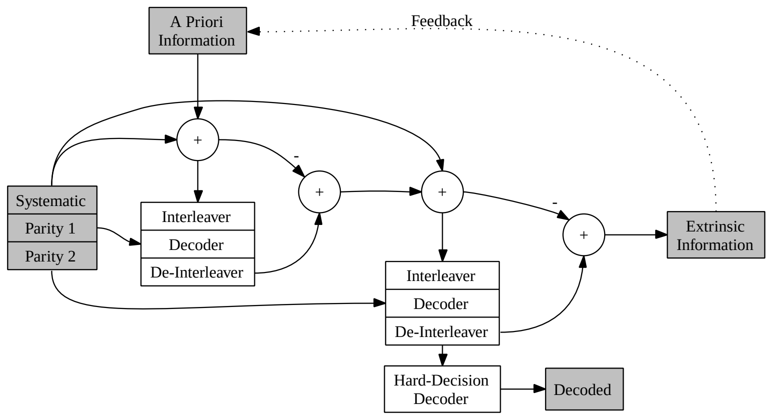


Figure: Structure of a Turbo Decoder

Non-Binary Turbo Codes

- How do we construct good q-ary turbo codes?
  - Literature in Binary
  - Almost nothing on non-binary convolutional or turbo codes

Turbo code results submitted to ISIT 2008:

- $GF(8)$ codes compared with triple-binary
  - Gao and Soleymani [5, 6]
  - Transmitted using 8-PSK Gray-mapped modulation
- $GF(16)$ codes
  - Transmitted using 16-PSK and 16-QAM Gray-mapped modulation
  - Overall spectral efficiencies of 1.33 and 0.8 bits/s/Hz
  - Compared with previous $GF(8)$ codes at 1 bit/s/Hz

8-ary code – bit error rate

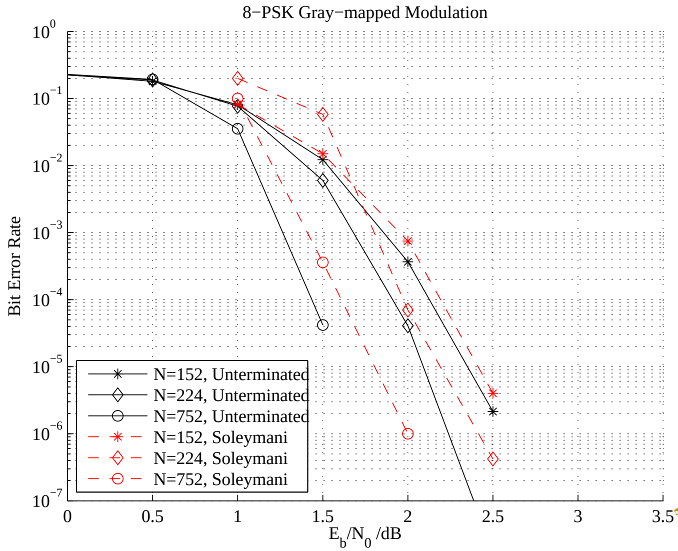


Figure: BER performance of our Turbo codes and Soleymani's

8-ary code – frame error rate

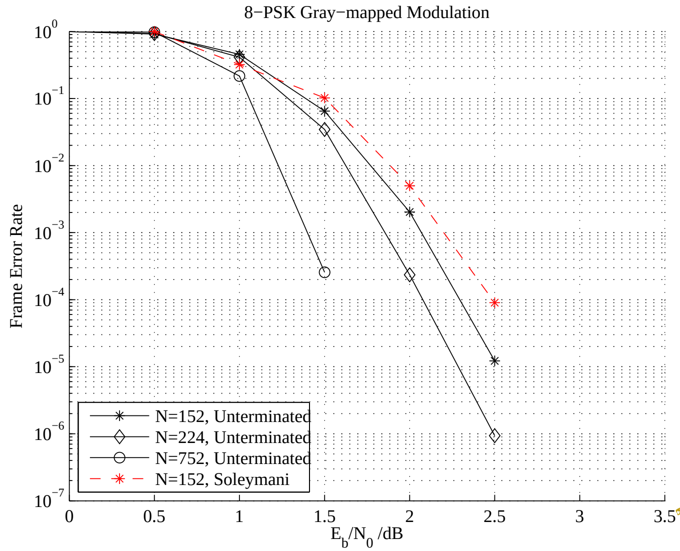


Figure: FER performance of our Turbo codes and Soleymani's

16-ary code – bit error rate

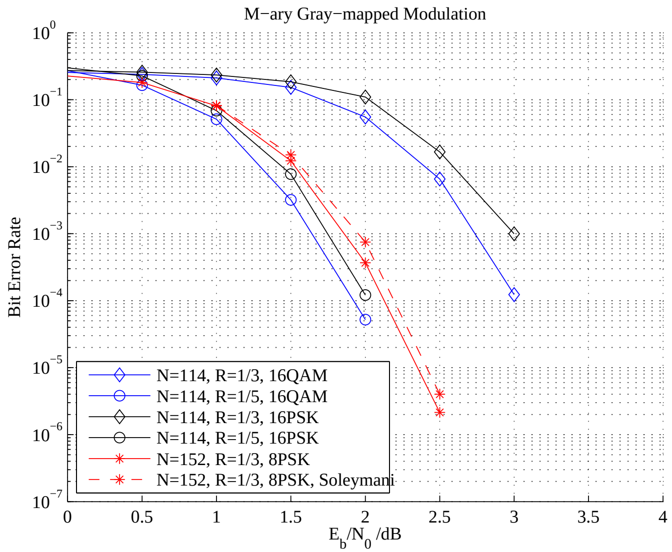


Figure: BER performance of 16-ary codes at close to 1 bit/s/Hz

16-ary code – frame error rate

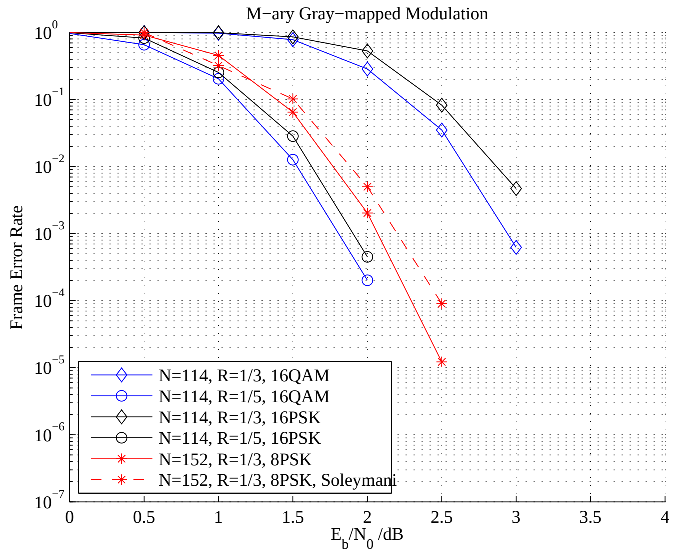


Figure: FER performance of 16-ary codes at close to 1 bit/s/Hz

16-ary code on q-SC

Symbol Error Rate

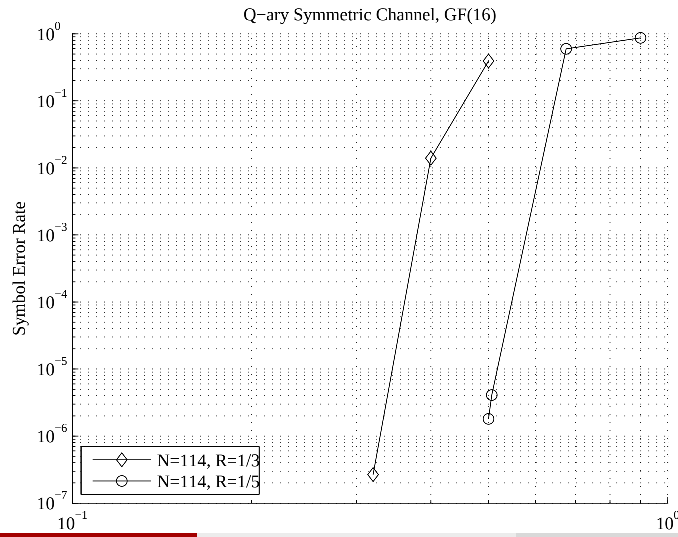


Figure:

16-ary code on q-SC

Bit Error Rate

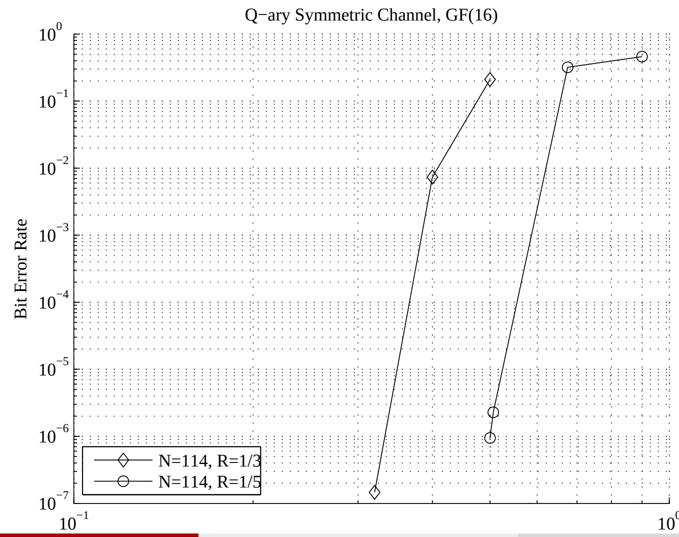
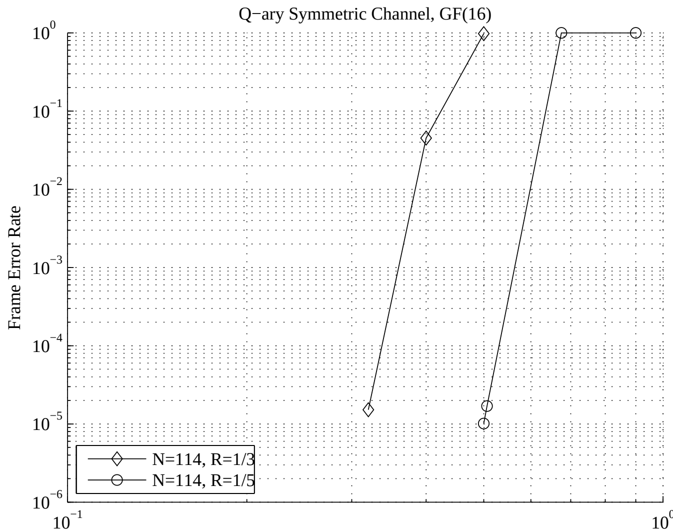


Figure:

16-ary code on q-SC

Frame Error Rate



Turbo-protected Watermark Codes

Hypothesis

We can replace the outer LDPC code used by Davey with a suitable Turbo code.

Experiment

- *For our purposes we need powerful codes*
- *Therefore we focus on low-rate systems*
- *We seek to compare with Davey's code I (rate $R = 1/20$); we construct:*
  - *Watermark code $k/n = 4/15$*
  - *Turbo code $GF(16)$, $R = 1/5$*
  - *Overall rate $R = 4/75$*

Turbo-protected Watermark Codes

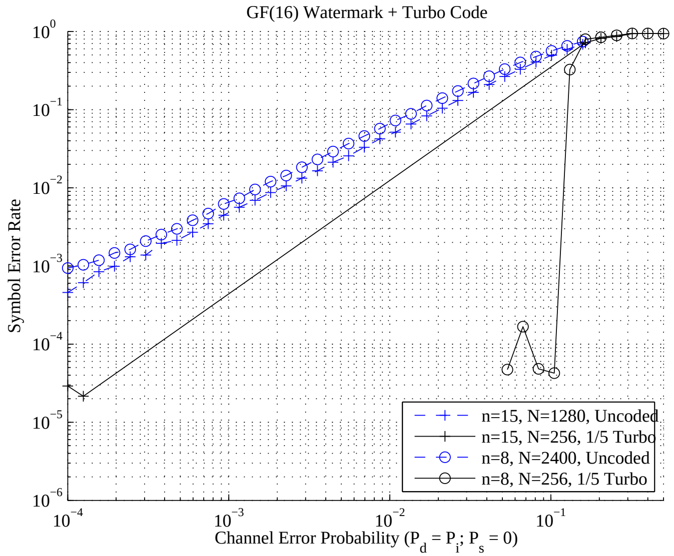


Figure: SER performance of Turbo-coded Watermark codes

Turbo-protected Watermark Codes

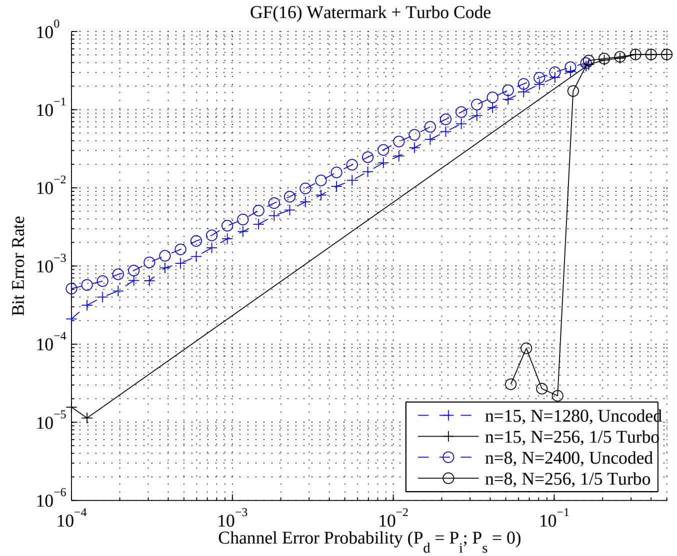


Figure: BER performance of Turbo-coded Watermark codes

Turbo-protected Watermark Codes

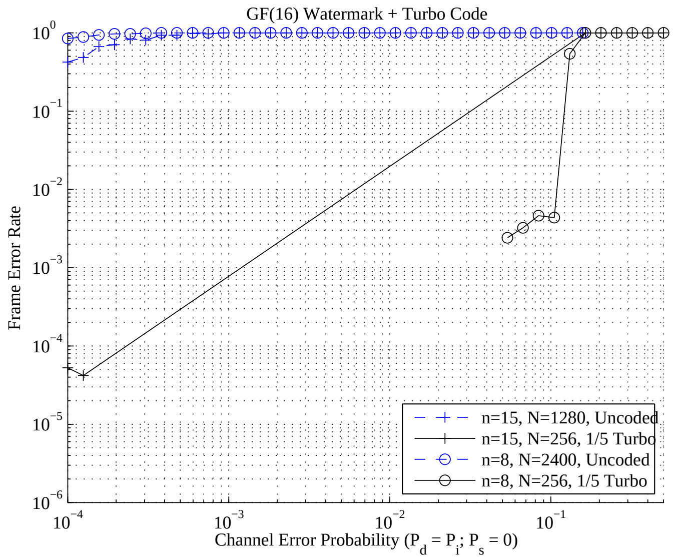


Figure: FER performance of Turbo-coded Watermark codes

Achievements

- We have improved on existing deletion/insertion correcting codes
- This has led us to introduce non-binary turbo codes
  - which have hardly been studied before
- This codes have good performance also on a *q*-ary symmetric channel (not surprising)

Open questions

- Optimality of underlying watermark codes
- Synchronisation correction in 2D
- Measure perceptual impact of the Stirmark attack

Questions?

T. Kalker, "Considerations on watermarking security," *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pp. 201–206, 2001.

L. J. Schulman and D. Zuckerman, "Asymptotically good codes correcting insertions, deletions, and transpositions," *IEEE Trans. Inform. Theory*, vol. 45, no. 7, pp. 2552–2557, 1999.

M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 687–698, 2001.

E. A. Ratzer, "Error-correction on non-standard communication channels," Ph.D. dissertation, University of Cambridge, 2003.

M. R. Soleymani, Y. Gao, and U. Vilaipornsawai, *Turbo Coding for Satellite and Wireless Communications*. Kluwer Academic Publishers, 2002.

Y. Gao and M. R. Soleymani, "Bandwidth-efficient transmission using nonbinary crsc turbo codes," *Electrical and Computer Engineering,*

2003. IEEE CCECE 2003. Canadian Conference on, vol. 3, pp. 1617–1620, 4-7 May 2003.