PERSEUS Library: New Challenges for an Alternative to Encryption

Eric Filiol
filiol@esiea.fr

ESIEA - Laval
Laboratoire de cryptologie et de virologie opérationnelles $(C+V)^O$

RMLL 2011

Introduction

- The protection of personal/private data is more than ever a critical and major issue.
  - Rise of private intelligence companies/agencies.
  - Evolution towards more and more controls on citizens for commercial (ex. : HADOPI), political (opponents, dissidents), economic (industrial espionage) . . . purposes.
- Almost all data flows/channels are now under monitoring and surveillance.
- Mobility (smartphone, laptop...) increases the risk dramatically.
- Very critical situation regarding non-democratic countries/dictatorship (Chine, Myanmar, Iran...) and even for some democratic ones (USA, UK...).

Introduction (2)

- Strong need for Nation State security at the same time as well as for their citizens (Police, Defense).
- The question is : how to prevent/avoid abuses regarding personal/private data monitoring/eavesdopping while
  - preserving Nation State ability to ensure Defense/Police missions (fight against terrorism, child pornography, homeland security, organized crime....)...
  - respecting the different existing national regulations regarding data protection techniques,
  - AND while preserving the natural and fundamental right of citizens to communicate freely and privately.
- One solution is the PERSEUS technology.

Introduction (3)

Classical not to say obvious solution : cryptography. However there are national regulations to protect and to ensure Nation State security. Moreover, the use of encrypiton is easily detected (lack of TRANSEC).

- To combine all those constraints (technical, legal...) we need
  - a concept which cannot be broken unless using a tremendous computing power during a significantly long time (typically a supercomputer during sevral days/weeks/months),
  - otherwise it cannot be broken in practice,
  - the use of this concept/technique must be difficult to detect (TRANSEC).
- This solution would naturally limit attempts of data eavesdropping abuses.
- The solution is to replace cryptography by coding techniques with controlled deterministic noise.

Introduction (4)

- Classical example : a western journalist in China.
  - He writes and sends his article from China.
  - The Chinese authorities can eventually break his messages (provided that they succeed in detecting them), but only after the journalist went back home in his country.
- Other cases : political opponents, decision-makers, CTO, CEO ...

Encryption vs Noisy Coding

- The "legal" definition of what is cryptography is in fact direclty connected to the following probability :

$$P[c_t = m_t + e_t] = P[e_t = 1]$$

where c_t, m_t are ciphertext and plaintest bits respectively and where e_t is the noise bit coming from the key and the encryption algorithm at time instant t.

  - If $P[e_t = 1] = \frac{1}{2} + \epsilon$ (with ϵ very close to 0) then it is cryptography.
  - Otherwise (ϵ significantly different from 0), it is noisy coding.

- The approach consists in considering a computationally untractable problem (for this attacker) taken from coding theory.

Encryption vs Noisy Coding (2)

- Why encoded data are far better than encrypted data :
  - Encrypted data have a maximal entropy profile that makes them easy to detect.
  - Noisy encoded data have on the contrary a low entropy profile that is quite close to that of plain data.
- This low entropy profile enables to bypass any detection techniques based on entropy analysis and statistical tests.
  - TRANSEC features are ensured by hiding our data in a traffic of same entropy.
  - Bypassing of Echelon-like filters, firewalls, IDS....

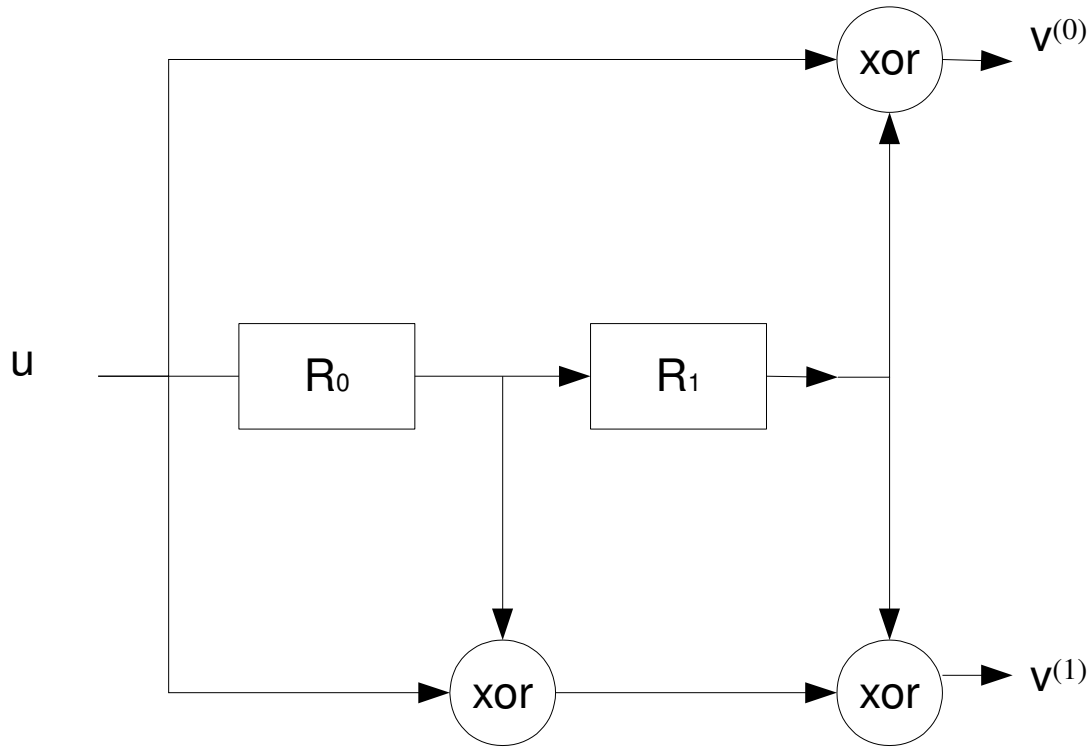
Plan

Error-correcting Codes

- PERSEUS is based on a widely used class of codes : punctured convolutional codes.
- Widely used in telecommunications, telephony (GSM, UMTS, GPRS...), satellites (turbo-codes)...
- Admissible noise rate : less than 1 %, decoding step being computationally complex and time/memory consuming.
- Except in very few cases (e.g Czech army) the codes used are known (public) and have relatively small values.

Convolutional Codes

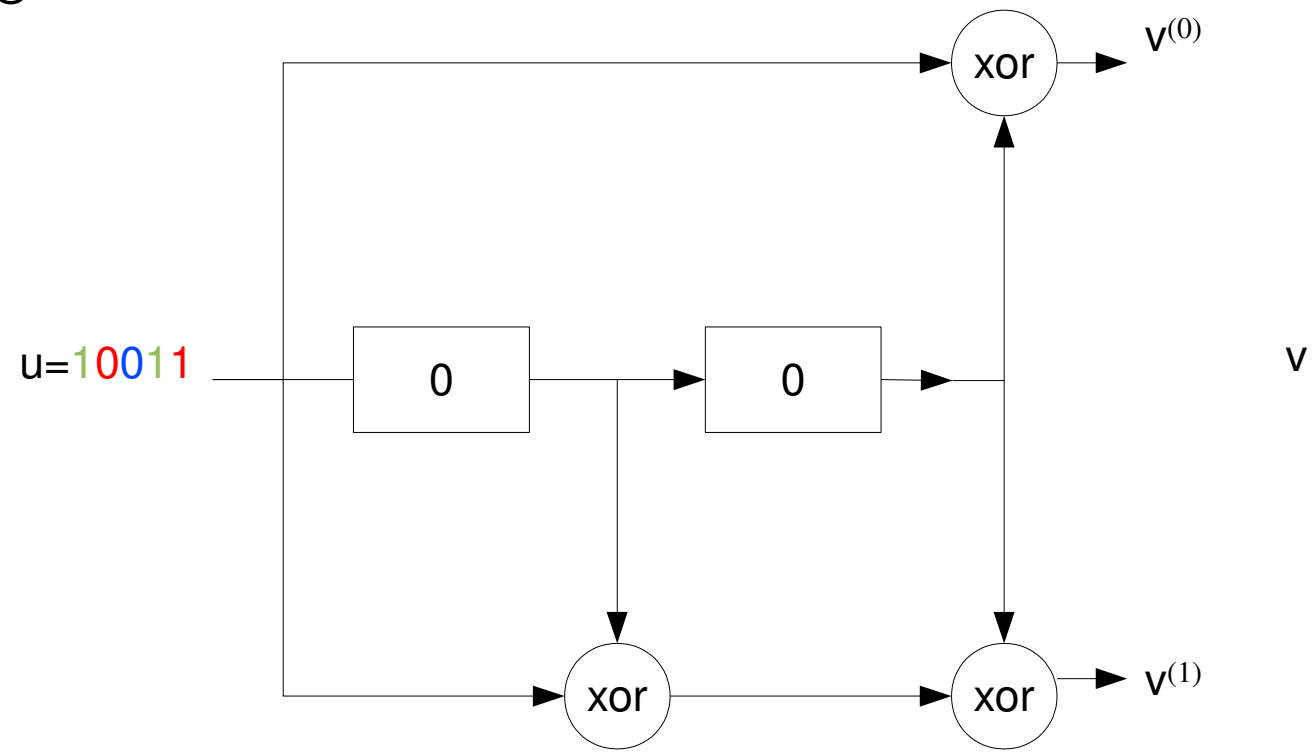
Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.



Convolutional Codes

Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.

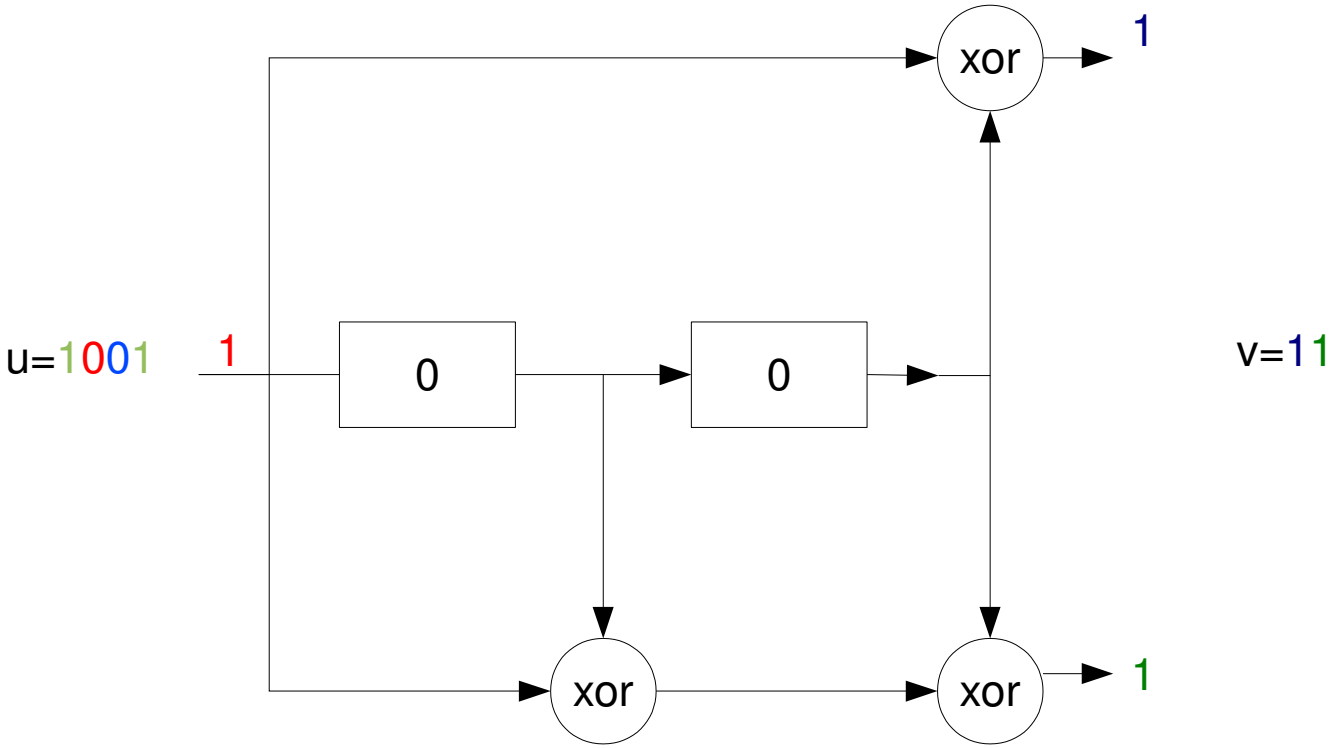
A message $u = 10011$



Convolutional Codes

Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.

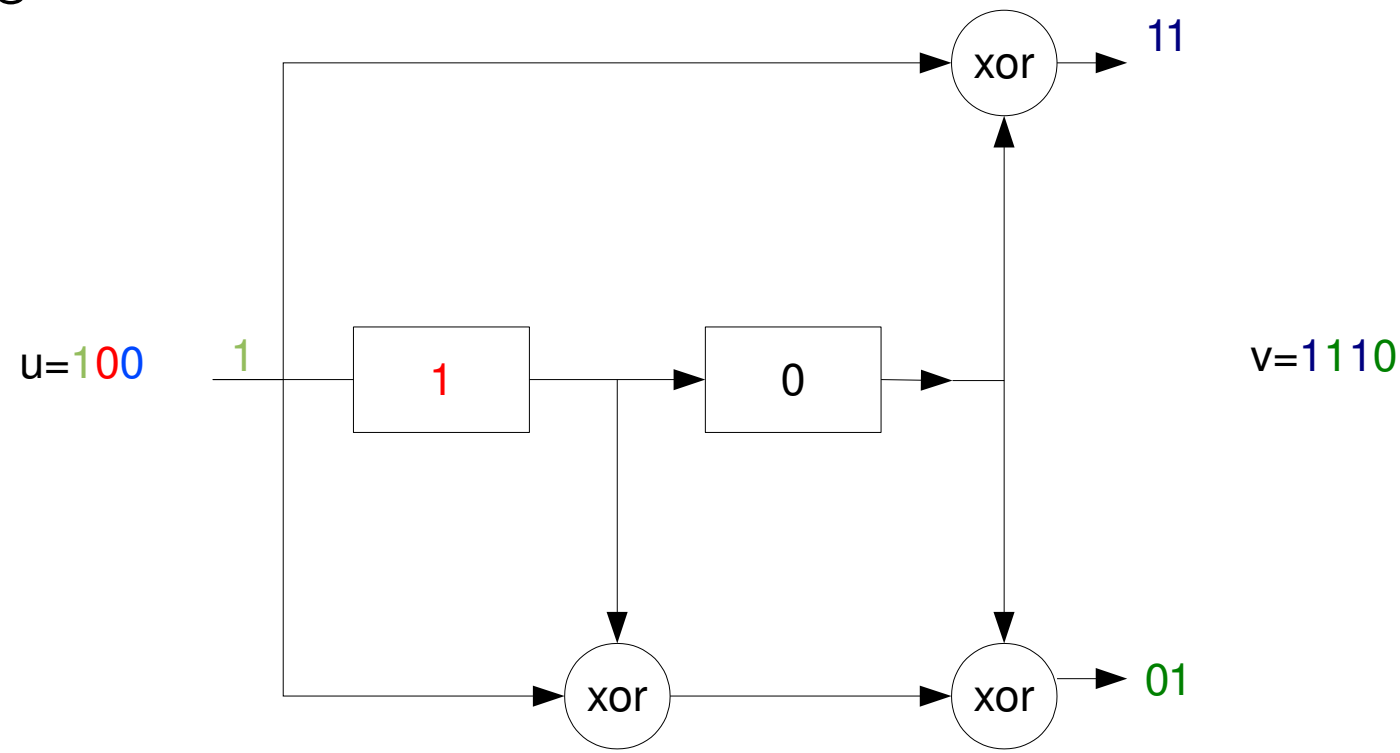
A message $u = 10011$



Convolutional Codes

Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.

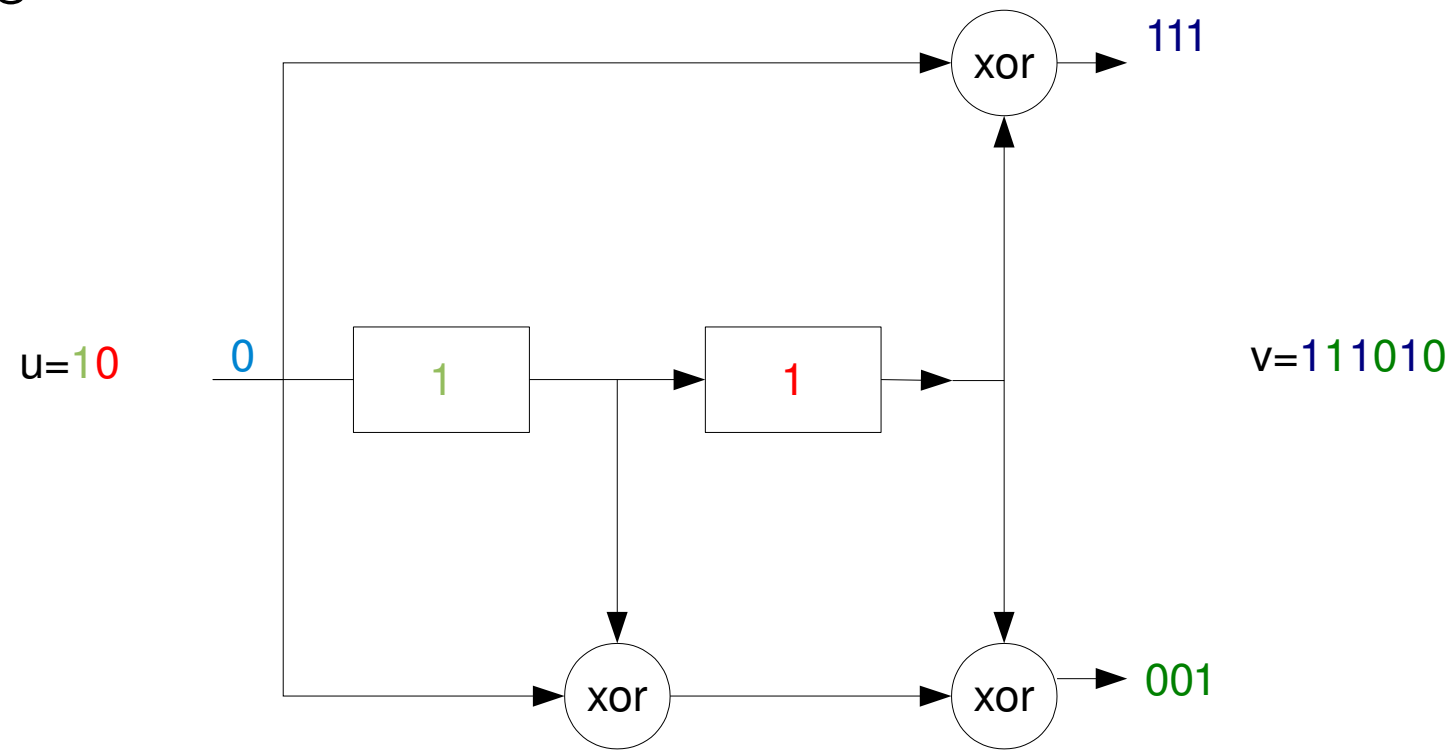
A message $u = 10011$



Convolutional Codes

Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.

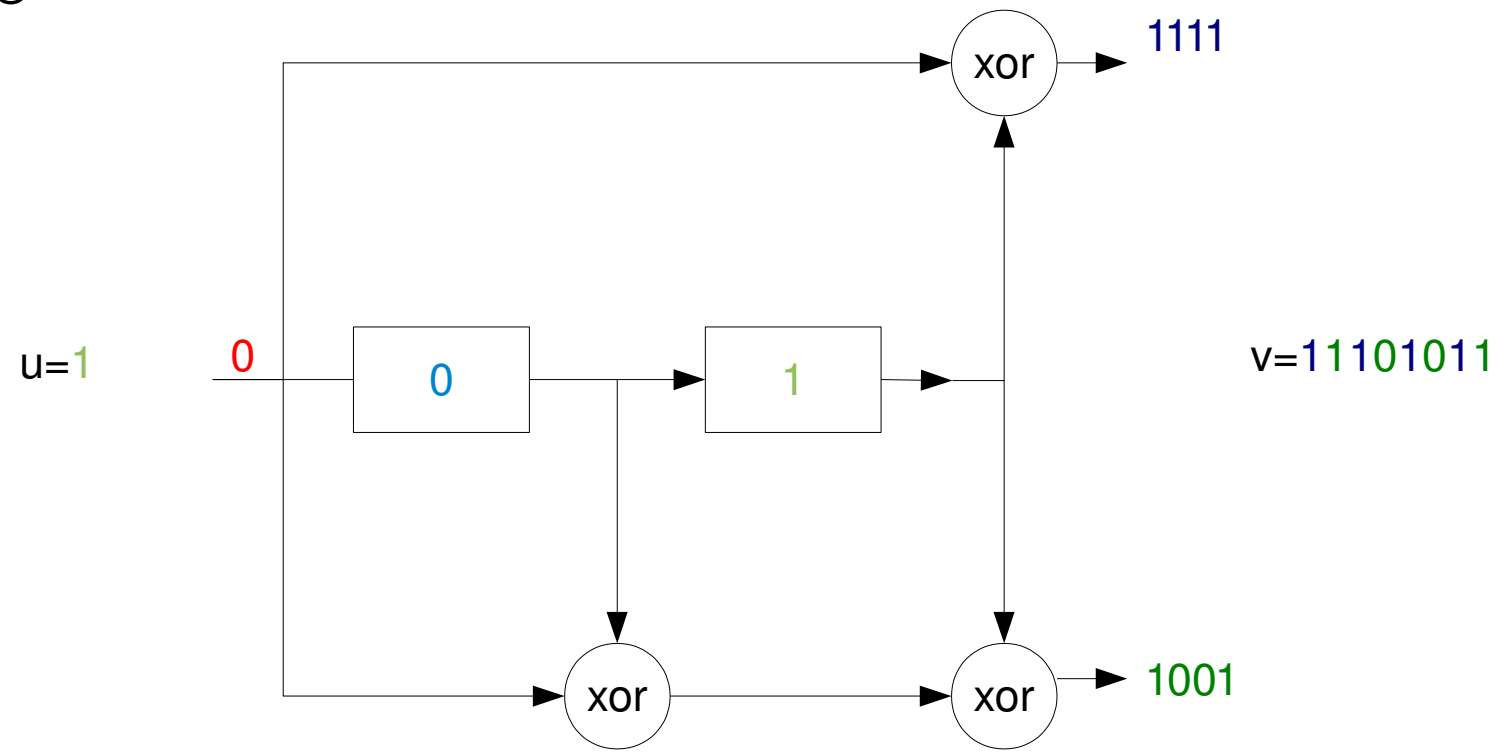
A message $u = 10011$



Convolutional Codes

Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.

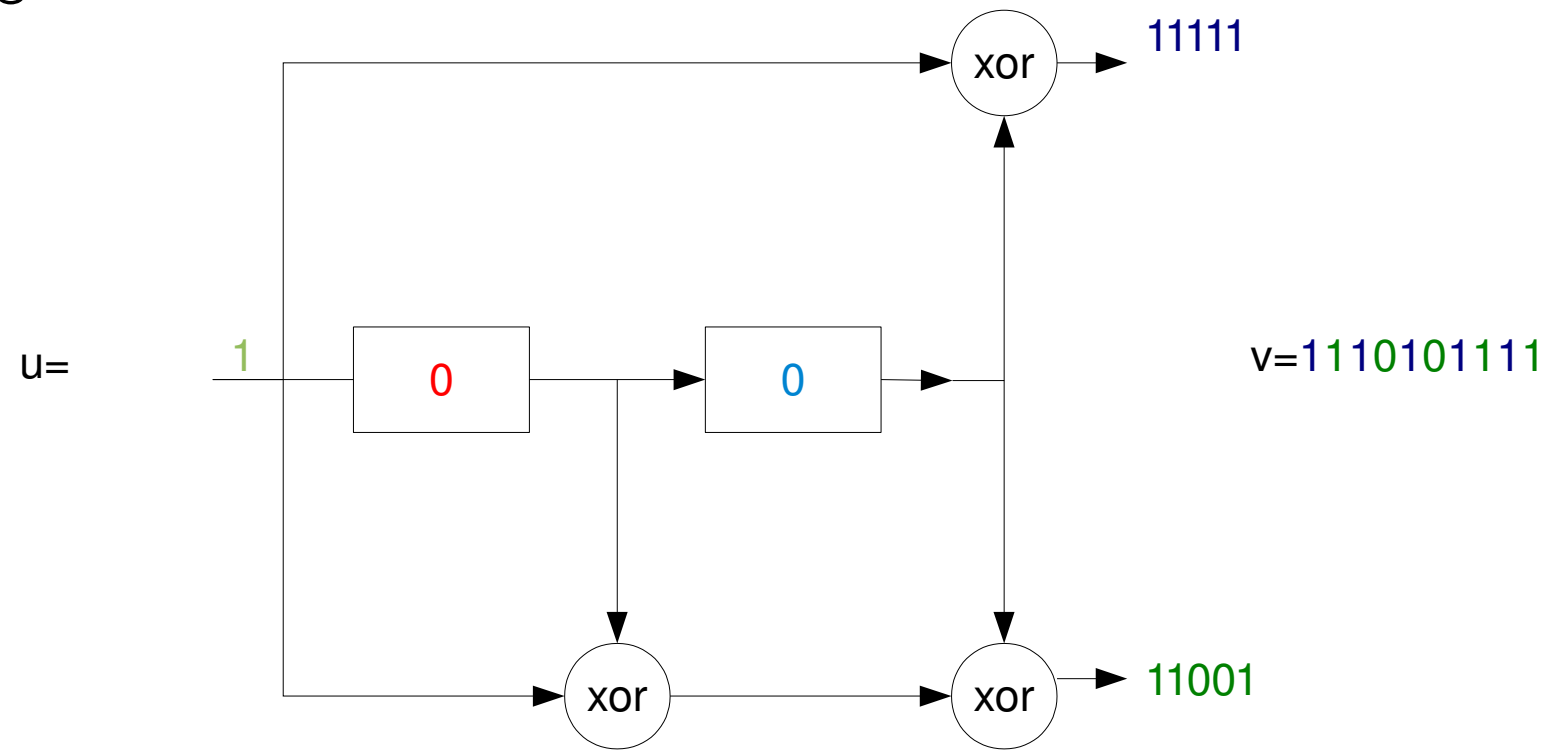
A message $u = 10011$



Convolutional Codes

Let $\mathcal{C}$ be a convolutional code of rate $\frac{1}{2}$ with a memory size (constraint) of $M = 2$.

A message $u = 10011$



Presentation

A convolutional code is defined by

- a rate : $\frac{k}{n}$
- a memory size (constraint length) $K = M + 1$.

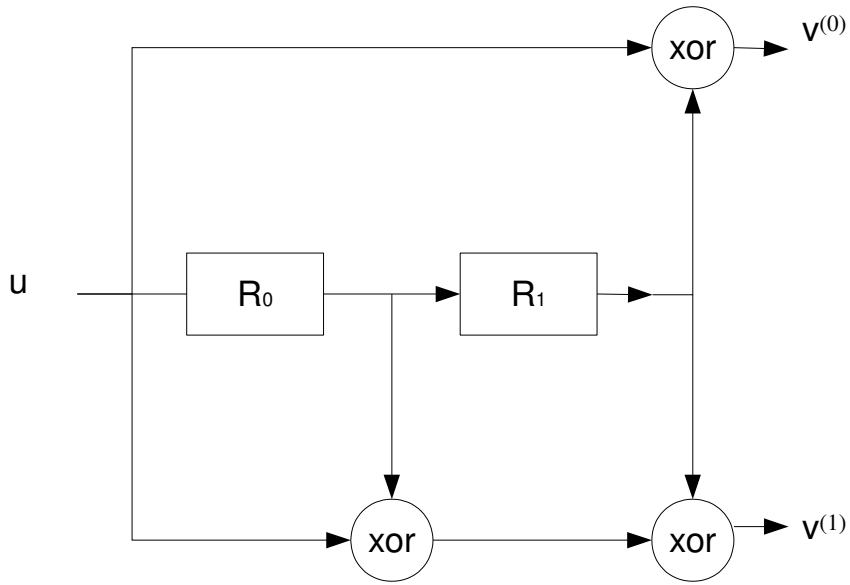
Notation

(n, k, K) code convolutif

Alternative Vision

Convolutional Code

k registers with n polynomials operating on each register.
$n \times k$ polynomials for a (n, k, K)-convolutional code.
The degree of polynomials will be equal to $K - 1$.



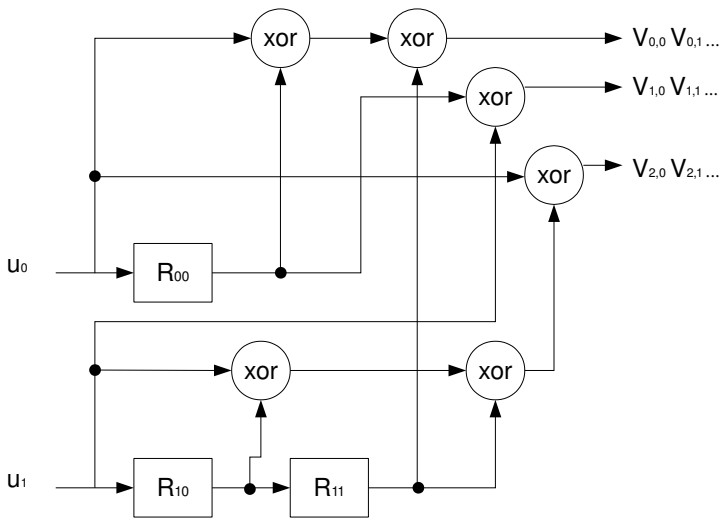
$\mathcal{C}$: $(2, 1, 3)$ code

$v_0 : 1 + x^2$
$v_1 : 1 + x + x^2$

Alternative Vision

Convolutional Code

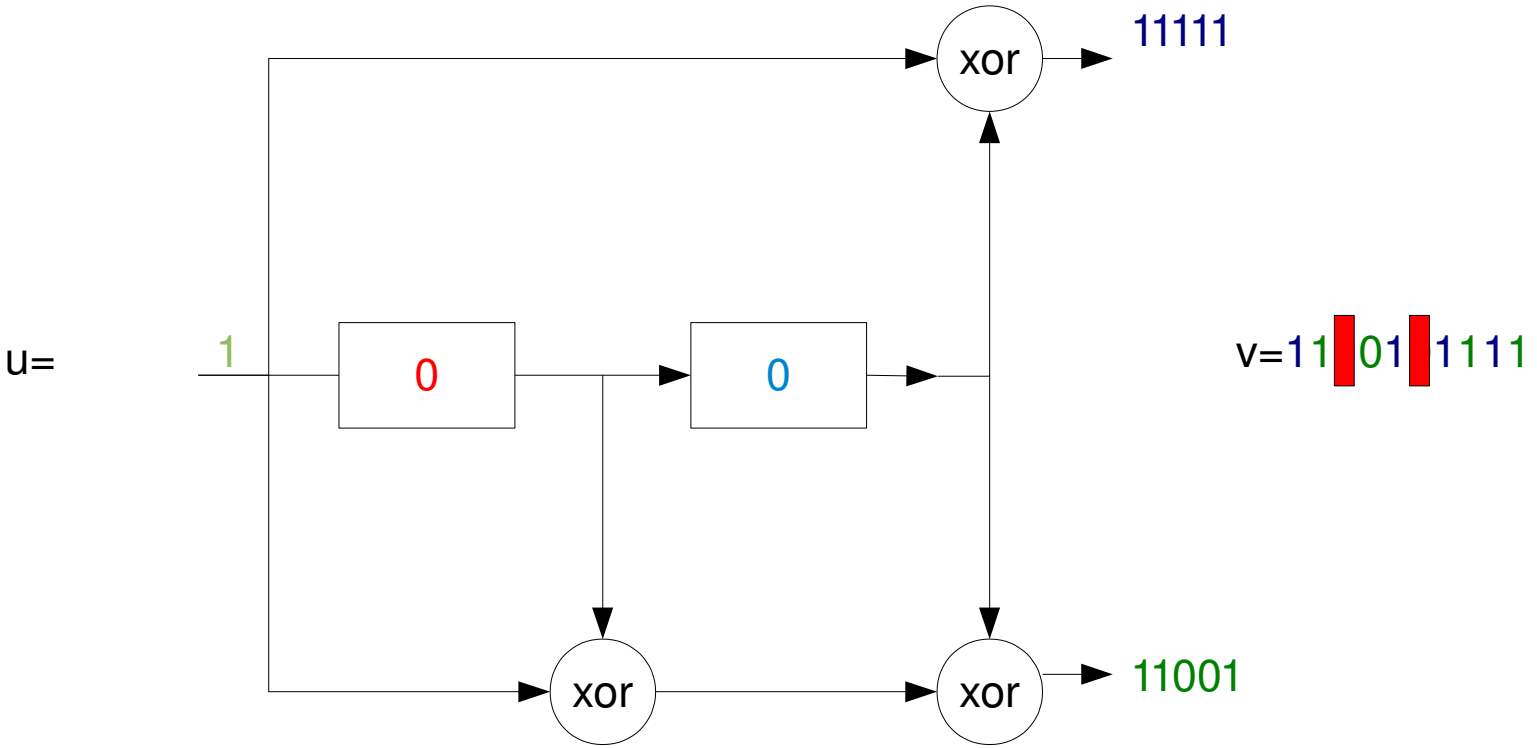
k registers with n polynomials operating on each register.
$n \times k$ polynomials for a (n, k, K)-convolutional code.
The degree of polynomials will be equal to $K - 1$.



$\mathcal{C}$: $(3, 2, 3)$ code

$v_{0,0} : 1 + x$
$v_{0,1} : x^2$
$v_{1,0} : x$
$v_{1,1} : 1$
$v_{2,0} : 1$
$v_{2,1} : 1 + x + x^2$

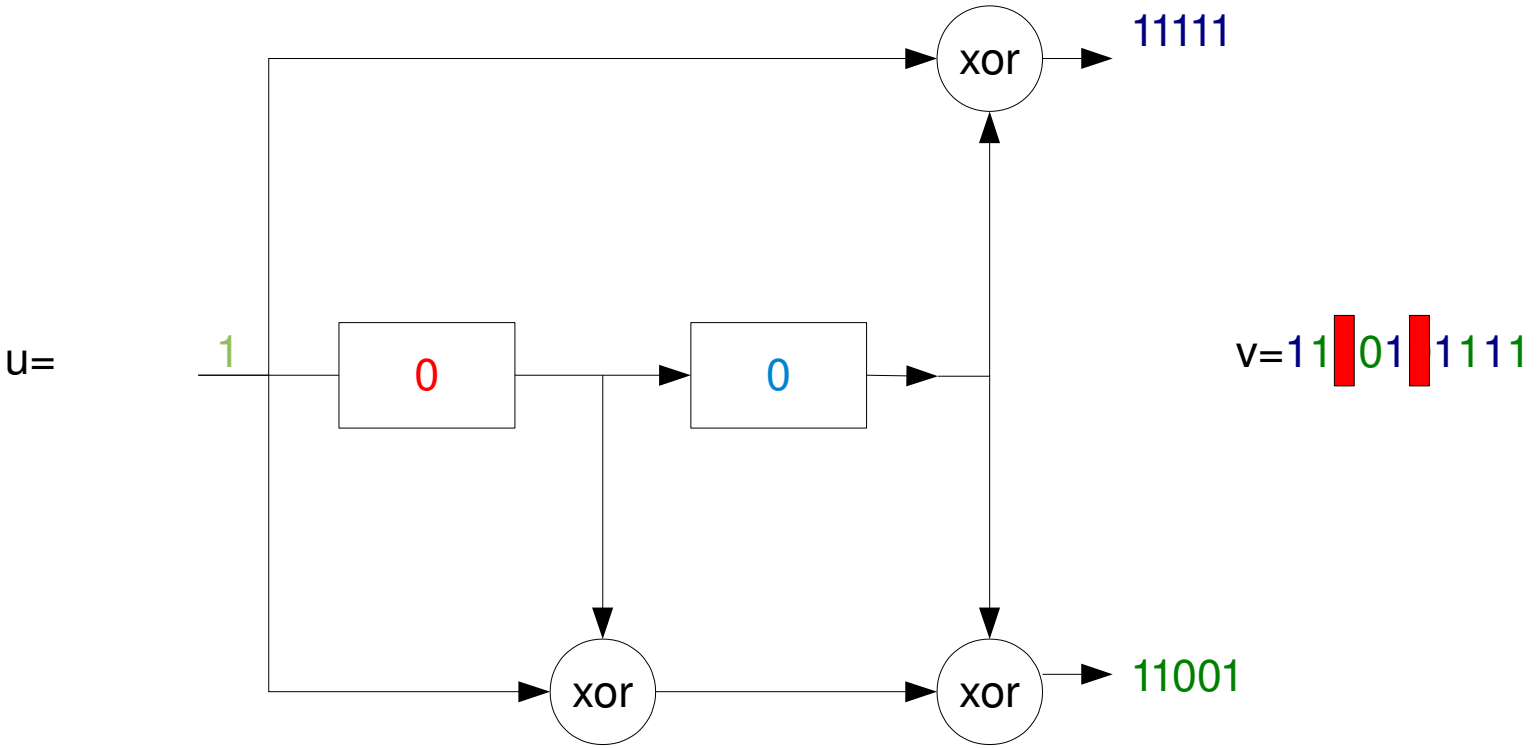
Puncturing



Puncturing pattern

P : a $J \times n$ matrix of weight I.

Puncturing



Puncturing pattern

P : a $J \times n$ matrix of weight I.

Exemple

Let P be the puncturing pattern given by :

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

and let v be the $(2, 1, 3)$ encoder output sequence :

$$v = \left(\begin{array}{cc|cc|c} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{array} \right)$$

$$\left(\begin{array}{cc|cc|c} 1 & & 1 & & 1 \\ 1 & 0 & 0 & 1 & 1 \end{array} \right) \Rightarrow 11010111$$

Why puncturing ?

1. Save bandwidth (reduce the redundancy added).
2. Produce an equivalent (non punctured) convolutional code which is stronger for our purposes (see further).

Why puncturing ?

1. Save bandwidth (reduce the redundancy added).
2. Produce an equivalent (non punctured) convolutional code which is stronger for our purposes (see further).

Equivalent (non punctured) convolutional code

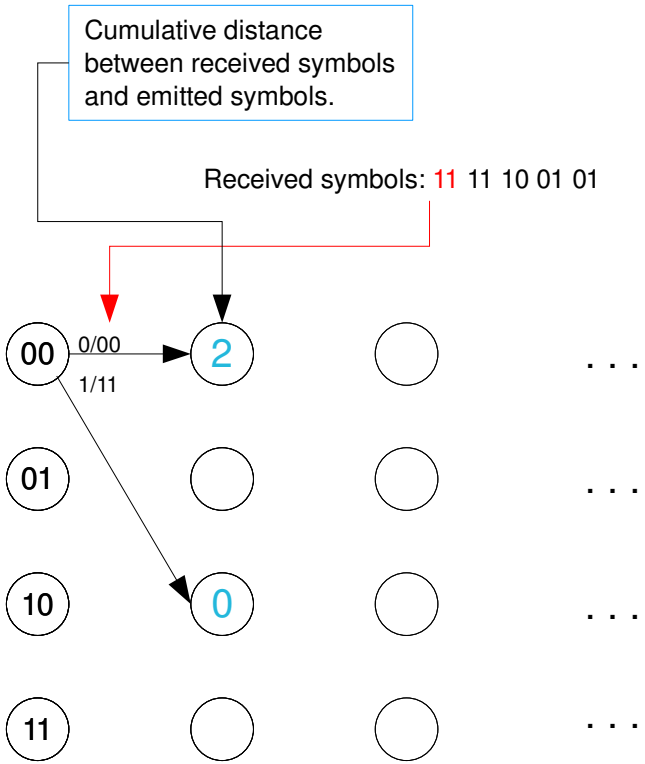
A (n, k, K)-convolutional code and a $J \times n$ puncturing matrix P of weight I.

$$\Rightarrow (I, kJ, K)\text{-convolutional code}$$

Viterbi Algorithm

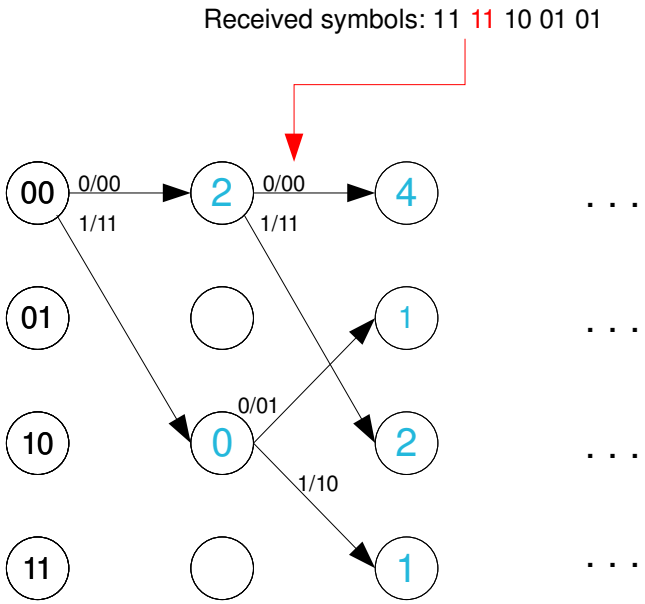
Viterbi Algorithm

Lattice construction



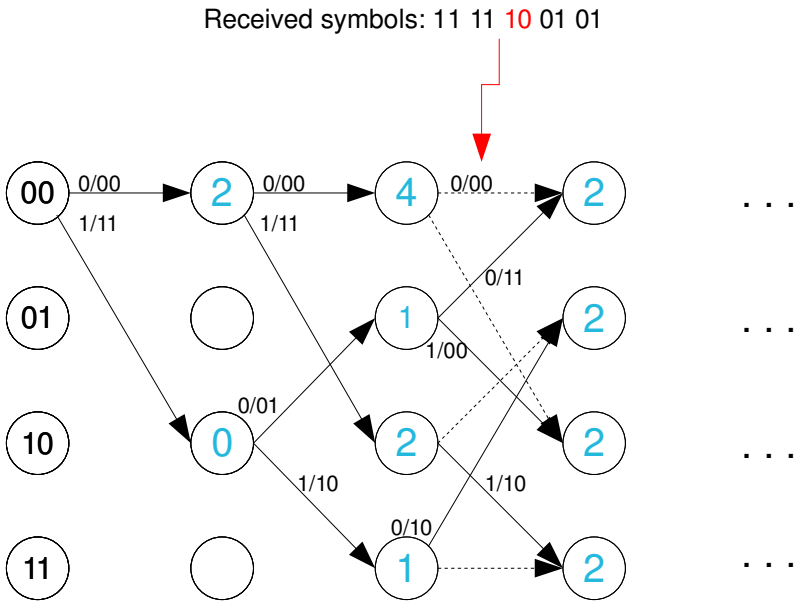
Viterbi Algorithm

Lattice construction



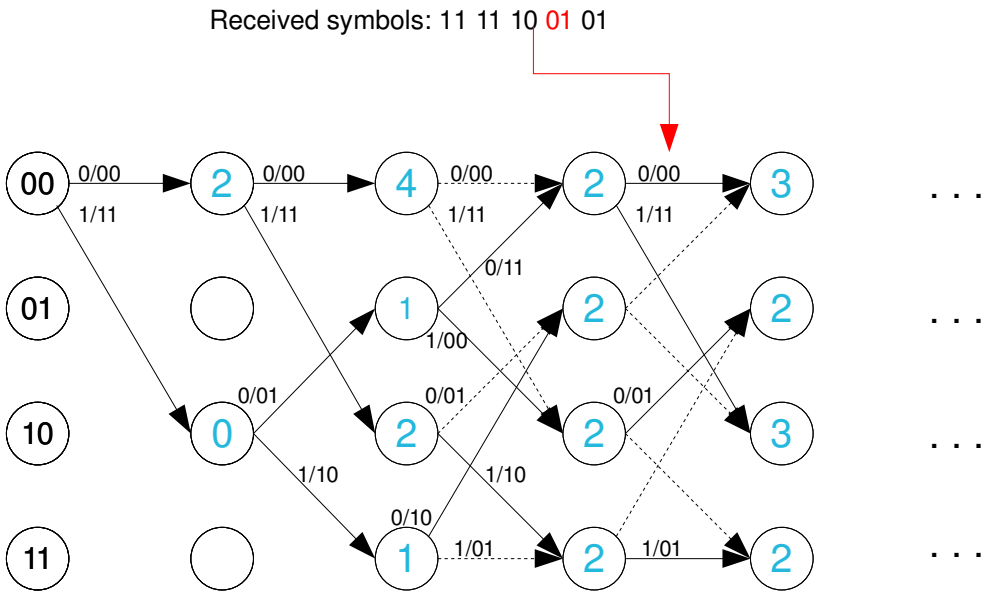
Viterbi Algorithm

Lattice construction



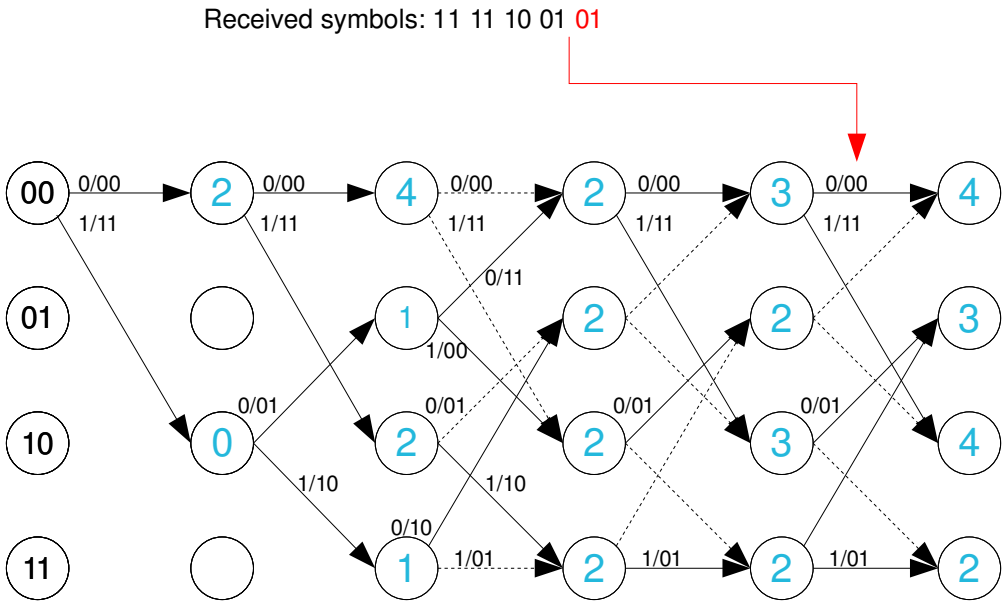
Viterbi Algorithm

Lattice construction



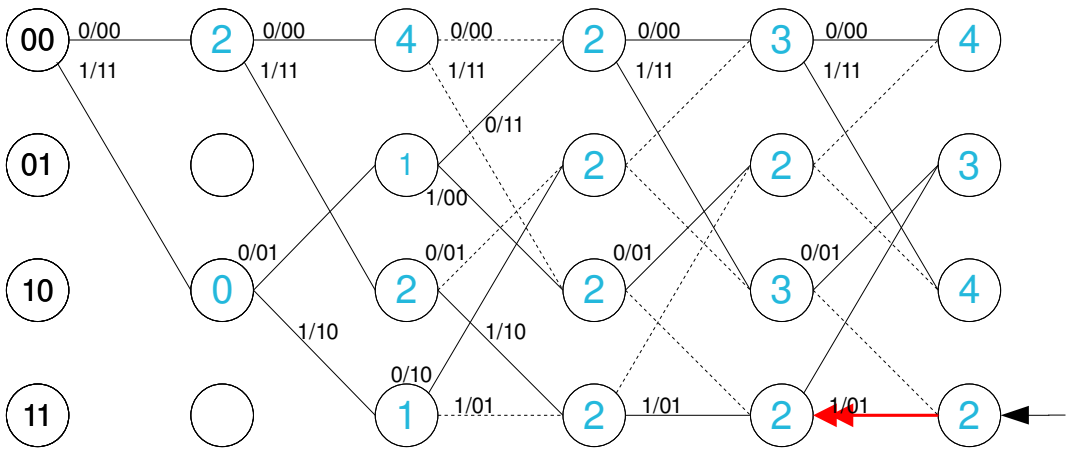
Viterbi Algorithm

Lattice construction



Viterbi Algorithm

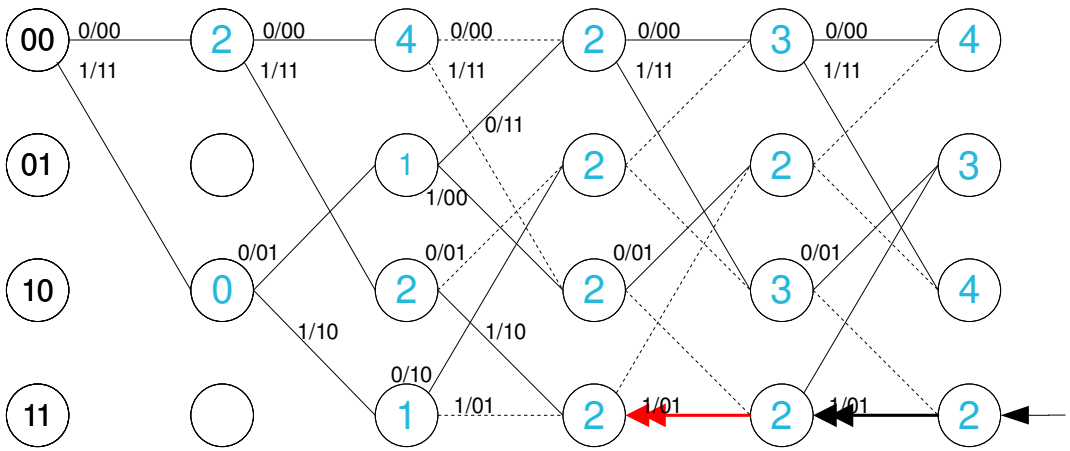
Backtracking



Decoded symbols: 1

Viterbi Algorithm

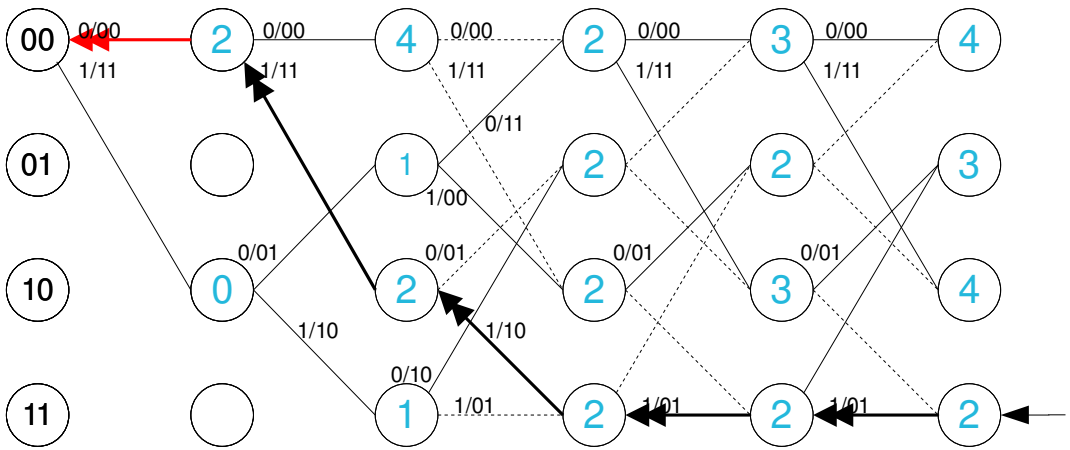
Backtracking



Decoded symbols: 11

Viterbi Algorithm

Backtracking



- Decoding has exponential complexity in K.
- When dealing with puncturing, replace removed bits with zeroes.

Convolutional Code Reconstruction (Filiol 1997 - Barbier 2007)

Aim : recovering all the parameters of an unknow encoder from the encoded data only, to be able to decode data afterwards.

Puncturing effect

Let us consider a (n, k, K)-convolutional code and a $J \times n$ puncturing matrix P of weight I :

$$\Rightarrow (I, kJ, K)\text{-convolutional code}$$

Reconstruction has the following complexity

$$\mathcal{O}(\alpha \times n^5 \times K^4) \Rightarrow \mathcal{O}(\alpha \times I^5 \times K^4)$$

α : grows exponentialy with p, the noise probability

Convolutional Code Reconstruction (Filiol 1997 - Barbier 2007)

Aim : recovering all the parameters of an unknow encoder from the encoded data only, to be able to decode data afterwards.

Puncturing effect

Let us consider a (n, k, K)-convolutional code and a $J \times n$ puncturing matrix P of weight I :

$$\Rightarrow (I, kJ, K)\text{-convolutional code}$$

Reconstruction has the following complexity

$$\mathcal{O}(\alpha \times n^5 \times K^4) \Rightarrow \mathcal{O}(\alpha \times I^5 \times K^4)$$

α : grows exponentialy with p, the noise probability

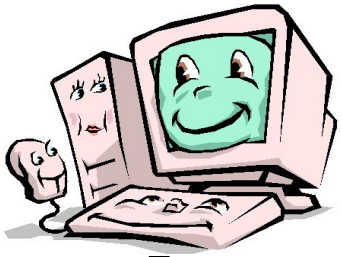
The Noise Impact

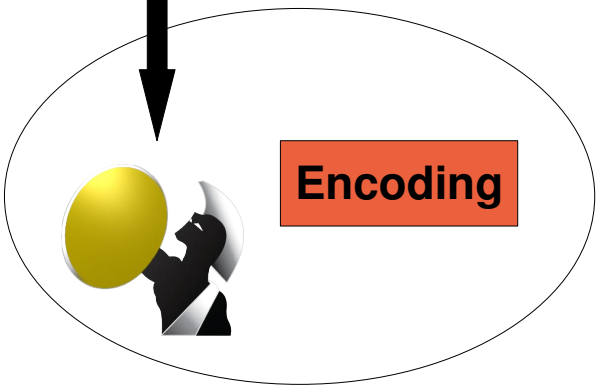
The probability to successfully reconstruct a code exponentially decreases with p. If $p > 10\% \Rightarrow$ online reconstruction is impossible ; offline reconstruction is computationally very hard.

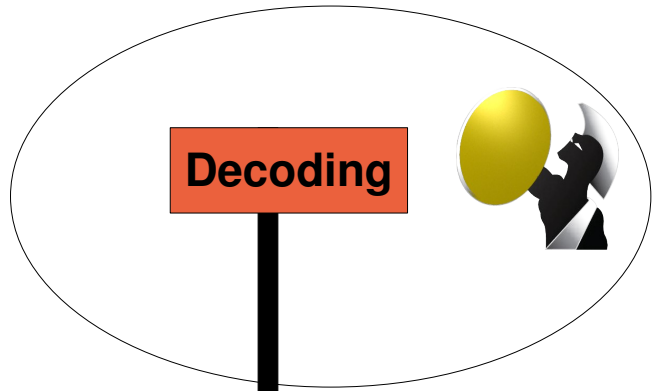
Encoder	Reconstruction Time ($p = 10^{-2}$)	Reconstruction Time ($p = 2.10^{-2}$)
(4, 3, 8)	7 min 12 sec	Failure
(4, 3, 9)	6 min 16 sec	Failure

TABLE: Examples of reconstruction times (Pentium IV 2.0 Ghz) for two levels of noise

Plan




Encoding


Decoding

General Principle

- The attacker must face a computationally untractable problem.
- The coder is changing randomly and frequently (every session, every frame...).
- To make the reconstruction computationally untractable we add a secret-based deterministic noise.
- The legitimate users only know the exact noise bit indices and can remove the noise to perform a noiseless sequence decoding.

Problem

Viterbi decoding easy as long as $p < 1 - 3\%$
Reconstruction is practically untractable as soon as $p > 5\%$ (inline mode)

General Principle

- The attacker must face a computationally untractable problem.
- The coder is changing randomly and frequently (every session, every frame...).
- To make the reconstruction computationally untractable we add a secret-based deterministic noise.
- The legitimate users only know the exact noise bit indices and can remove the noise to perform a noiseless sequence decoding.

Problem

Viterbi decoding easy as long as $p < 1 - 3\%$
Reconstruction is practically untractable as soon as $p > 5\%$ (inline mode)

Solution

We add a random, deterministic noise with $p \in [10\%, 30\%]$

For every new session or frame

1. $5 < n \leq 12$
2. $1 < k < 6$
3. $10 < N \leq 30$
4. $n \times k$ polynomials of degree $N - 1$
5. A $J \times n$-matrix P of weight $(n \times J) - (J - 1)$
6. X_0 a 128-bit value (initialization of the noise generator).

For every new session or frame

1. $5 < n \leq 12$
2. $1 < k < 6$
3. $10 < N \leq 30$
4. $n \times k$ polynomials of degree $N - 1$
5. A $J \times n$-matrix P of weight $(n \times J) - (J - 1)$
6. X_0 a 128-bit value (initialization of the noise generator).

For every new session or frame

1. $5 < n \leq 12$
2. $1 < k < 6$
3. $10 < N \leq 30$
4. $n \times k$ polynomials of degree $N - 1$
5. A $J \times n$-matrix P of weight $(n \times J) - (J - 1)$
6. X_0 a 128-bit value (initialization of the noise generator).

Parameter Management and Protection

Protected through a common secret quantity (key) or a cryptographic protocol (initial HTTPS session).

Implementation

- Written in C (relatively optimized to remain readable).
- Version 1.x : with Viterbi decoding.
- Triple licence MPL/GPL/LGPL.
- Source, documentation available on `http://code.google.com/p/libperseus`
- Bugs, feedbacks and comments are welcome (`ffiliol@gmail.com`).

Structure of the Library

Very simple structure : 5 main procedures.

- int Gen_Pcc(PUNCT_CONC_CODE *);
- int Gen_Noise_Generator(NOISE_GEN *, INIT_NOISE_GEN *);
- int Gen_Noise(unsigned char *, NOISE_GEN *, unsigned long int, INIT_NOISE_GEN *);
- int PCC_Encode(unsigned char *, unsigned char *, PUNCT_CONC_CODE *, unsigned long int);
- int Viterbi_Decode(unsigned char *, unsigned char *, PUNCT_CONC_CODE *, unsigned long int);

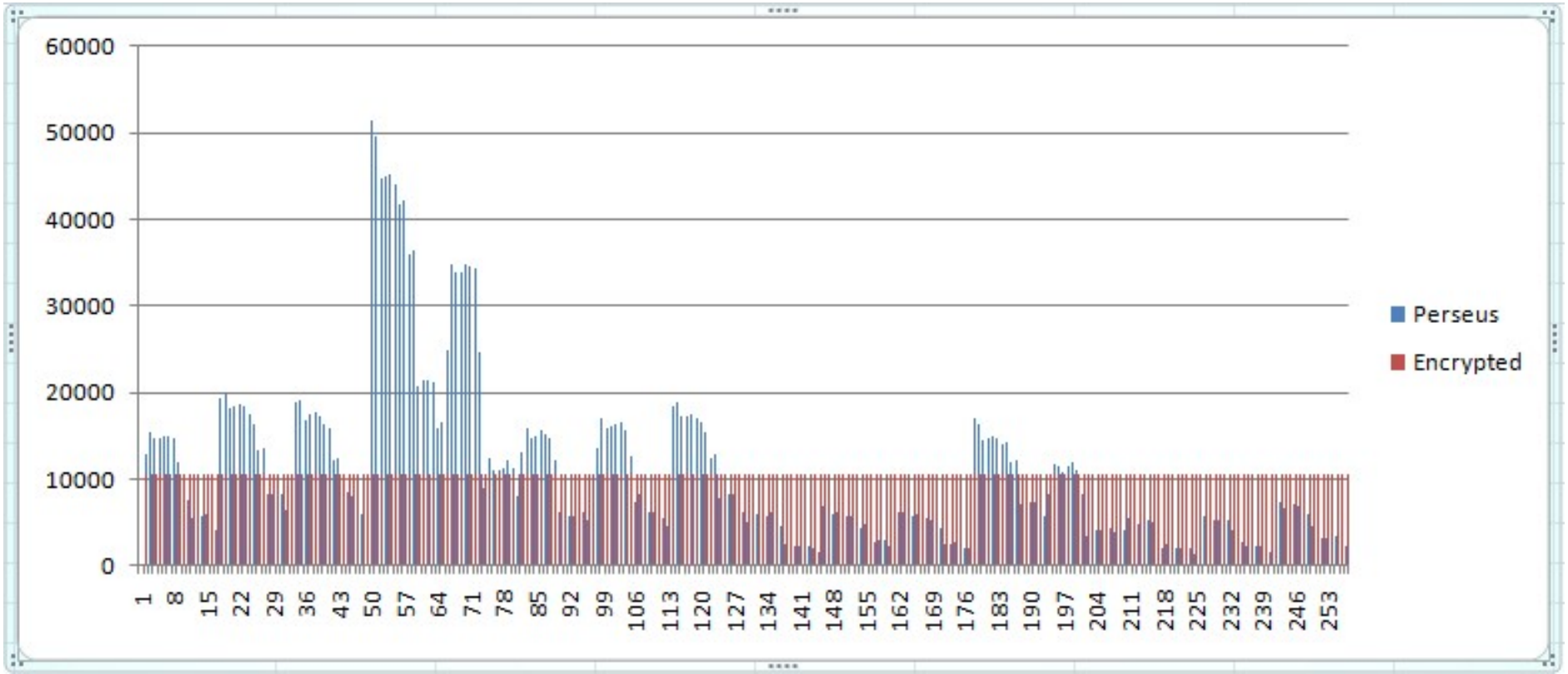
TRANSEC Aspect

ny PERSEUS-protected stream exhibits a statistical profile that is far from that of any encrypted data.

Noise probability	Plain data average entropy	PERSEUS-protected data data	Encrypted data
5 %	4.21	4.96	8.00
10 %	4.21	6.19	8.00
15 %	4.21	6.46	8.00
20 %	4.21	7.11	8.00
25 %	4.21	7.39	8.00
30 %	4.21	7.45	8.00
35 %	4.21	7.71	8.00

TABLE: Average entropy profile for plain, PERSEUS-protected and AES encrypted data

TRANSEC Aspect



TRANSEC Aspect

- The PERSEUS traffic is polymorphic by nature since we change the encoder very frequently and randomly.
- It is not possible to distinguish a particular user.
- Development of the module MIMIC to simulate and mimick any fixed traffic or data,
  - At the present time, it is non public.
  - Legal aspects to clarify.

Roadmap 2008 - 2011

- Conception, formalization and technical validation of the concept (Eric Filiol).
- Firefox Plug-in to protect HTTP traffics (Eddy Deligne).
- Andromeda library to protect TORRENT traffics (Fabien Jobin)
- PERSEUS library version 1.x (Eric Filiol)
  - Version 1.6 soon (code cleaning, last bugs fixing) ;
- OpenBSD port (Pierre-Emmanuel André).
- Application for off-line protection of files (Jonathan Dechaux et Eric Filiol) : release at end of July 2011 (currently under final tests/code checking phases).
- Industrial support from DFT-Technologies (`http://www.dft-techno.com`).

Roadmap 2011 - 2013

- Version 2.x of the library :
  - Polynomial time decoding.
  - Parallelization support (OpenMP).
- VoIP protection.
- Libre Office (cloud version) security.
- Video & streaming protection.
- Android Application (SMS, Voice...).
- USB/Ethernet, Ethernet/Ethernet sticks fro mobile environment protection.
- MIMIC module ? ? ?

Conclusion

- PERSEUS technology gives an elegant answer to a critical issue :
  - How to protect against HTTP traffic eavesdropping by botnet clients...
  - ... and abuses against citizens'privacy fundamental rights...
  - ... without crypto...
  - while preserving TRUE, LEGITIMATE ability for national security enforcement (internal and external) ?
- Until now, more than 200,000 downloads (all applications).
- Many contacts, feedbacks, comments received. Thanks to all who helped.

Contributers & Thanks

- Pierre-Emmanuel André (port OpenBSD), Jonathan Dechaux (off-line application), Eddy Deligne (plug-in Firefox, port Ruby), Guillaume Delugré (security analyse of implementation), Anthony Desnos (port Python), Fabien Jobin (Andromeda).
- Ltc Frédéric Suel (DGSIC), Libre Office fundation.
- Bowman Wangeci (BAE Systems).
- Thanks to all who helped/contributed but who want to remain anonymous.
- Thanks to all who use PERSEUS.

References

- Plug-in Firefox `http://code.google.com/p/perseus-firefox`
- Librairie Andromède `http://code.google.com/p/andromeda`
- Librairie Perseus `http://code.google.com/p/libperseus`
- Eric Filiol (2010). "PERSEUS *Technology : New Trends in Information and Communication Security*". Article Open Access `http://arxiv.org/abs/1101.0057`.

Questions ?