平成24年度修士論文

Churn and Tampering Resilience in Network Coding-based Anonymous P2P Network

ネットワーク符号化に基づく

匿名P2PネットワークのChurnとTampering耐性

大学院情報システム学研究科

情報ネットワークシステム学専攻

学籍番号	:	1052034
氏名	:	Huszák Gergely György
主任指導教員	:	森田　啓義　教授
指導教員	:	笠井　裕之　准教授
指導教員	:	小川　朋宏　准教授
提出年月日	:	平成24年7月20日

Abstract of Master's Thesis

研究科・専攻	大学院　　　　　　　　学研究科　　　　　　　　　　専攻 UEC, Graduate School of Information Systems, NS		博士前期課程 Master's Course
氏　　　　名	Gergely György Huszák	学籍番号	1052034
論 文 題 目	Churn and Tampering Resilience in Network Coding-based Anonymous P2P Network		

Abstract

This Master's Thesis proposes an extension to an existing Anonymous Peer-to-peer network, to make it resilient to churn and tampering.
The thesis starts by presenting the existing Network Coding-based Anonymous Peer-to-peer Network of Wang et al. that provides advantages over earlier system from the viewpoints of computational efficiency and level of anonymity provided. That system consists of well-known and -researched building blocks, such as Network Coding and Onion Routing, but it combines them in a novel way, so that the resulting design is capable of providing high level of sender side confidentiality.
Subsequent part of the thesis presents the problem at hand being the existing scheme's lack of resilience against common error situations, such as failure of node software or hardware, network congestion etc. The importance of a network – where the number of messages belonging to a single anonymous session is quadratic to the basic system parameters – being error prone through a well-established mechanism is demonstrated and explained through examples of how decoding is carried out in the present scheme.
Taking advantage of some of the mechanisms already present in the system today (such as Threshold Secret Sharing Scheme, XOR etc.) and of the locality of the effect of node errors, the thesis proposes a double-redundant coding technique that can make the system resilient against communication failure, as long as the number of node failures is at most the arbitrarily chosen integer r (that stands for redundancy).
Subsequently the thesis points out some of the desirable corollary of the presented coding technique, such as being able to withstand a complete main path failure at no additional costs or difficulty of decoding. Next the thesis presents an evaluation of the relationship between the strength of churn resilience and offered level of anonymity through simulations. During this evaluation process the thesis reconstructs the simulation method used by the original work and then establishes its own – wider-spectrum – attack model along which subsequent conclusions are drawn.
Based on the outcome of this process, the thesis points out that there is a trade-off between the two properties of churn resilience and anonymity and presents some guidelines along which the necessary level of both may be achieved.
Further along the way the work proposes an improvement over the coding-scheme initially presented. The improvement aims for advancing the network economical aspect (total amount of data traveling through the mesh) of the final scheme through replacing the inner-code by a systematic Reed-Solomon Code, as an Erasure Code.
Building on the properties of the final scheme, the thesis demonstrates an easy extension of the newly created technique to make the network able to recover not only from errors of random nature (churn), but also from malicious nodes' purposeful and deliberate action towards disrupting network communication, a feature referred to as Tampering Resilience.
Finally the thesis points out some aspects not discussed in the original design, such as the excessive combinatorial and computational complexity that cooperating attackers need to face when trying to compromise anonymity, and highlights several paths that can be taken in the future to further improve the advantageous properties of the newly presented scheme.

修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　　　　　学研究科　　　　　専攻　博士前期課程 UEC IS/NS 情報システム学研究科 情報ネットワーク　専攻　博士前期課程		
氏　　　名	Huszák Gergely György フサーク　ゲルゲイ	学籍番号	1052034
論 文 題 目	Churn and Tampering Resilience in Network Coding-based Anonymous P2P Network ネットワーク符号化に基づく匿名 P2P ネットワークの Churn と Tampering 耐性		

要　旨

論文は，既存のネットワーク符号化法に基づく匿名 P2P ネットワークに，Churn 耐性ならびに耐タンパ性をもたらす拡張法について論じる．
　まず最初に，Wang らによって提案された匿名性 P2P ネットワークについて，計算効率と匿名性の強さの観点から紹介する．
　Wang らのシステムは，すでによく知られたネットワーク符号化とオニオン・ルーティングを新しく組み合わせることによって，送信側における高いレベルの匿名性を可能にしている．
 次に，Wang らの匿名性 P2P システムでは，ソフトあるいはハードの故障やネットワークの輻輳といった，ネットワーク上ではよく起こりうる誤りに対する耐性が欠けるという問題点を指摘する．
　彼らのシステムでは，一つの匿名セッションで送信されるメッセージ数はシステムの基本パラメータの２次オーダになるため，送信誤りが発生しやすい傾向にあることを明らかにし，さらに，いくつかの例を用いて，どのようにメッセージが復号されるかについても説明する．
　本論文では，Wang らの匿名性 P2P システムの中で用いられている，いくつかの手法，例えば，しきい値秘密分散法や XOR 演算など，ならびに，ノード障害は局所的にしか影響を及ぼさないことを利用して，新たに，ノード障害に対する耐性を匿名 P2P ネットワークにもたらす二重符号化法を提案する．
　そして，提案符号化法では，新たにコストを増やすことなく，また容易に，ノード障害に対する耐性を実現できることを示す．
　さらに，本論文では，組合せ論に基づく理論的な手法と，シミュレーションに基づく実践的な手法を用いて，Churn 耐性の強さと匿名性レベルとの関係について吟味する．
　これらの関係を調べるため，本論文では，Wang らの論文で示されているシミュレーション結果を追試した上で，より汎用的な，ネットワークの攻撃モデルを新たに確立する．
　これらの結果より，本論文では，Churn 耐性と匿名性には二律背反の関係があることを示し，さらに，両者を満足いくレベルに保つためのガイドラインを与える．
　さらに，提案する二重符号化法をより改良するための方法についても考察する．
　二重符号化で用いる内符号をリード・ソロモン符号を用いて，シンボル消失誤り訂正を行うことによって，最終的なシステムのネットワーク効率を促進する．
　提案システムの特性を活かすことにより，本論文では，提案システムを，悪意をもったノードがネットワーク通信を途絶させることを意図した攻撃からネットワークを復旧できる，いわゆる，耐タンパ性をもつシステムへ，容易に拡張できることを示す．
　最後に，本論文では，最初の設計では検討されていなかったいくつかの観点，たとえば，複数の攻撃者が協同して匿名性を暴くために必要な計算量などを指摘し，提案手法の利点をさらに改良するいくつかの方策を提示する．

Churn and Tampering Resilience in Network Coding-based Anonymous P2P Network

ネットワーク符号化に基づく匿名P2PネットワークのChurnとTampering耐性

by

Gergely Huszák

フサーク　ゲルゲイ

Master's Thesis

修士論文

Department of Information Network Systems

Graduate School of Information Systems

The University of Electro-Communications

Tokyo, Japan

電気通信大学, 東京

July 2012

Table of Contents

1 List of Special Notations

$\ell, m, n, r, k, x, y, z, p, q, i, f$	Lower case italic letters: Scalars
$\mathcal{K}, \mathcal{N}, \mathcal{D}$	Capital cursive letters: Scalars
$\widetilde{n}, \widetilde{k}, \widetilde{s}$	Italic letters with tilde: Code parameters
S, D, R, O	Capital italic letters: Nodes
$\mathbf{M}, \mathbf{m}, \mathbf{v}, \mathbf{c}, \mathbf{r}$	Bold letters: Vectors (binary or symbolic)
$\mathbf{m}^{(x)}$	Upper index in parentheses: Stage index
$\mathbf{m}_{1..m}$	Dotted index: Abbreviation for enumeration Example is equivalent to $\mathbf{m}_1, \mathbf{m}_2,\ ...,\ \mathbf{m}_m$ May be used along with other notations, e.g. $\mathbf{c}_{y,x,1..m}$
$\widehat{\mathbf{m}}$	Bold letters with hat: Estimators
$\|\|$	Two vertical bars: Concatenation (of vectors)
$\|\|_{z=1}^{m}\ \mathbf{m}_z$	Concatenation sign with 2 indices: Iterative concatenation of vectors Example is equivalent to $\mathbf{m}_1 \parallel \mathbf{m}_2 \parallel\ ... \parallel \mathbf{m}_m$
$\mathbf{m} \oplus \mathbf{c}$	Circle with an equally sized plus sign in it: Binary bit-wise XOR operation between equally sized vectors $\mathbf{m}$ and $\mathbf{c}$
$\oplus_{z=1}^{m}\mathbf{c}_{z,x,y}$	XOR sign with indices: Iterative XOR of equally sized vectors Example is equivalent to $\mathbf{c}_{1,x,y} \oplus \mathbf{c}_{2,x,y} \oplus ... \oplus \mathbf{c}_{m,x,y}$
iff	If and only if (necessary and sufficient condition)
$a \cdot b$	Small dot: Scalar multiplication ($a \cdot b \equiv ab$)
$\mathcal{O}(n^c)$	Sets collecting problems or other sets with the same complexity or cardinality c (respectively)

2 List of Abbreviations

AC	Anonymous Communication
AC-ITCN	Anonymous Communication via Information Transmission based on Network Coding
AP2P(C/N)	Anonymous Peer-to-Peer (Communication/Network)
CA	Cooperating Attacker
CR	Churn Resilience
CRC	Cyclic Redundancy Check
EC	Erasure Code
ECC	Error Correcting Code
ES	Existing Scheme [2]
HF	Hashing Function
IT	Information Theory
MDS(C)	Maximum Distance Separable (Code)
MT	Master's Thesis
NC	Network Coding
NM	Network Message
P2P	Peer-to-Peer
P2PC	Peer-to-Peer Communication
P2PN	Peer-to-Peer Network
PS	Proposed Scheme
RI	Routing Information
RSC	Reed-Solomon Code
RSSS	Ramp Secret Sharing Scheme
RV	Random Vector
SSS	Secret Sharing Scheme
SSSS	Shamir's Secret Sharing Scheme
TR	Tampering Resilience
TSSS	Threshold Secret Sharing Scheme
XOR	Exclusive Or

3 List of Figures

4 Introduction

4.1 Background

The last decades have witnessed considerable development in the fields of **Peer-to-Peer Communication** (P2PC) and **Networks** (P2PN) [13]. The primary target of such solutions is to offer information sharing services between peers without the need for any special – e.g. centralized – entities. Most typically this is achieved by the members of the network themselves offering relaying services to each other: a node may be an endpoint in one and then act as a network element in an other session. The term *peer* comes from the fact that participating nodes have equal rights and roles in the network.

One of the most notable advantage of such systems is **robustness**: both node failures and attacker's actions have limited influence on the overall system's availability. One of the most notable item on the flip side of the coin is the difficulty of guaranteeing minimum level and quality of service, as there usually is a big diversity between the capabilities of the peers and in a purely P2P system none of the entities can influence the participating node beyond the possibilities provided by the underlying mechanisms and protocols equally available for each participant.

Anonymous Peer-to-Peer Communication (AP2PC) is built on the concept of P2PC, while stepping beyond it by providing mechanisms to **hide the true identities** of the sender, the receiver or both of a given session. Starting from the initial solutions (Anonymizer [3], Napster [9] etc.) the evolution of the anonymous solutions has been steady and provided several working solution (Chaumian networks [17], Onion

Routing [4], Tor [16], Tarzan [15], Crowds [6] etc.) previously and currently being available publicly.

An **Anonymous Peer-to-Peer Network** (AP2PN) may be categorized from multiple perspectives, such as:

1. Direction of communication:
   - **One-way**: Similar to the traditional simplex communication
   - **Two-way**: Using the above analogy, this works as a duplex channel. The actual communication taking place between the endpoints is up to the higher layers: may be send/acknowledge- or request/response-type
2. Scope of anonymity: This may be **sender-**, **receiver-** or **mutual-anonymity**.
3. Minimum features required to be possessed by an attacker to gain information and/or control:
   1. Impersonating relay nodes may be sufficient
   2. Need to be able to snoop on all lines of the network

AP2PN may provide additional features, such as search- or query-services, or those might be outside of the scope of the anonymous part of the system (e.g. may be web-based or centralized etc.)

One of the recent works of Wang et al. [2] has proposed a novel technique to establish **one-way sender-anonymity** that – they claim – provides elevated level of anonymity (increased robustness against attacker) compared to other existing solutions [14]. That work demonstrates a construction technique that relies on lightweight Network Coding (NC) [10][11] scheme to realize anonymous data flow from sender S to final destination D.

4.2 Problem Description

While the proposed scheme [2] does seem to advantages over earlier solutions for the same problem, it does does not seem to address the topic of **churn resilience**. In certain systems this shall not be a concern, but due to the fact that the proposed scheme possesses the following properties, we believe it is imperative to explicitly deal with **churn** and related issues:

1. The communication is **uni-directional**: acknowledgement-based resending is not possible
2. **Number of nodes** participating in a transfer **is large** ($\in \mathcal{O}(n^3)$) and the messages sent between them is even larger ($\in \mathcal{O}(n^4)$)
3. **Each and every** node and the messages sent by them is absolutely necessary to provide successful transfer from S to D

4.3 Overview of Content

This Maser's Thesis (MT) will address the following items:

1. Detailed **explanation of the Existing** Anonymous Communication via Information Transmission based on Network Coding (AT-ITCN) **Scheme** (ES) [2] and demonstration of its weakness
2. **Presentation of our proposed solution** that implements **Churn Resilience** (CR) and detailed explanation of its mechanisms
3. Discussion of the network-economy aspects of such a system and proposing improvements for it using an ideal **Erasure Codes** (EC)
4. Demonstration of a construction technique that makes and actual **Reed-Solomon Code** (RSC) – used as an EC – suitable for our scheme
5. Discussion of the relationship between **CR and anonymity** via simulations
6. Comparison of our results to that of the original paper [2]
7. Possibility of using our newly established scheme to implement **Tampering Resilience** (TR)
8. Final conclusions on our proposed scheme

9. Discussion of topics, such as what other improvements of the original design may be possible through further researc

5 The Existing Scheme

5.1 Introduction

The ES achieves one-way AC from sender S to final destination D through the following 4 steps carried out by S:

1. S selects ℓ **relay nodes** $R_{1..\ell}$
2. It sets up ℓ mesh networks to be able to reach each R anonymously
3. S *sends* ***routing information*** (RI) (also referred to as Path Setup Information) *to each* R using these pre-constructed meshes
4. Finally it sends the actual secret via $R_{1..\ell}$ to D using traditional **Onion Routing** (OR) [4]

Step 3. is *highlighted* because this is the step where most of the novelty of our work lies. As explained above, our network is two-fold and Fig. 1 shows the complete network, with the outer-layer at the top and the inner-layers at the bottom.

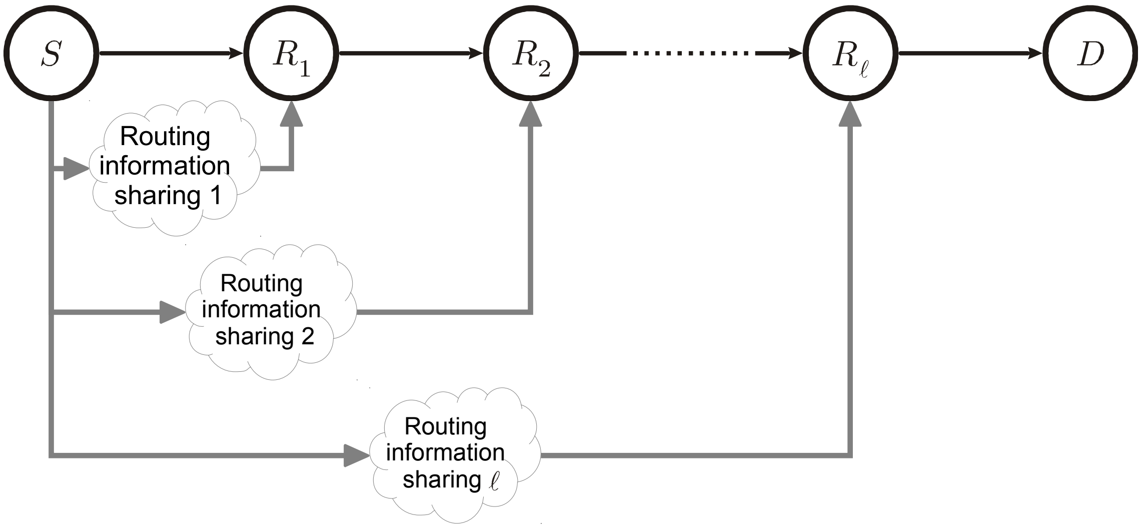

S
R_1
R_2
R_ℓ
D
Routing information sharing 1
Routing information sharing 2
Routing information sharing ℓ

Fig. 1: Top-level (bird-eye) view of the complete network

From among the 4 steps listed above, step 1. is represented by the intermediate OR relay nodes R, step 2. manifests itself through the network clouds (this is the inner-layer of our network), step 3. is represented by the gray lines (ending in bigger arrows) and finally step 4. is shown by the black lines (having smaller arrows).

Up till now we have talked about exclusively the outer-layer of our network, which relies on traditional techniques known from OR, while the inner-layer holds most of the features the ES is notable for. At step 2. S has to share some information with each R so that during the OR-phase, these relays can properly route the secret. The details of this information is not discussed, but – due to the fact that the paper [2] explicitly mentions *traditional onion routing* [4] – we assume that RI includes network (Layer 3) addresses of the next relay node along with one-time symmetric encryption keys to make those able to "peel-off" one layer of the onion.

5.2 Distribution Technique for RI

The way this (step 3. in chapter 5.1) is achieved is through the following means:

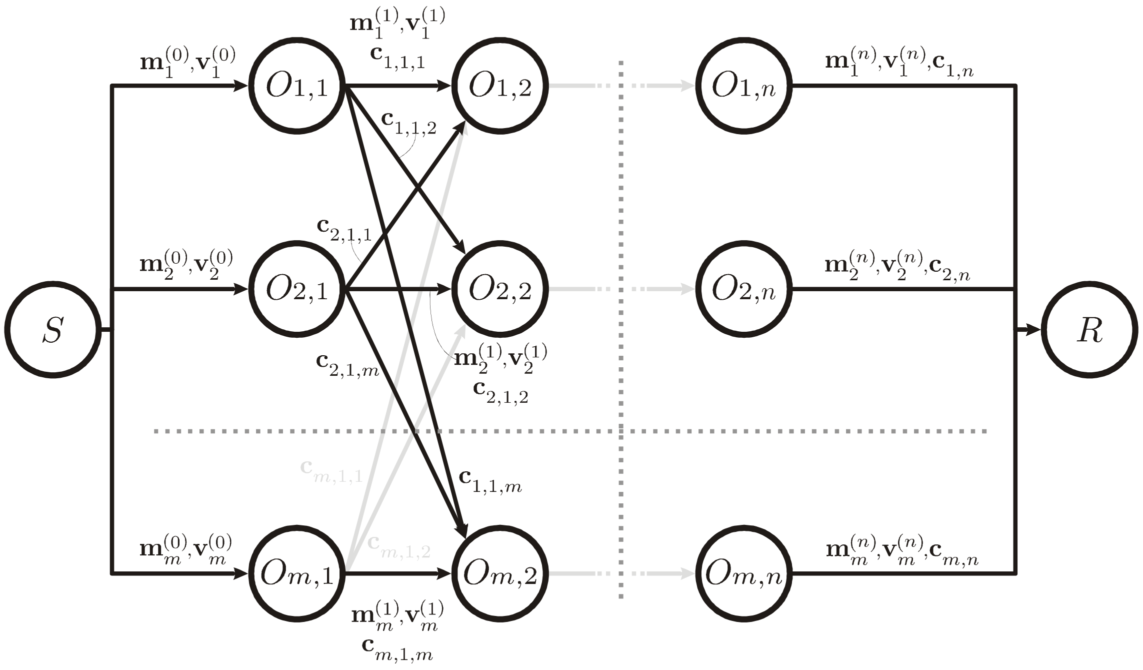
1. S uses a coding technique to divide the RI (from here referred to as $\mathbf{M}$) into m equally sized slices ($\mathbf{m}_{1..m}$) – in a way that **only possessing all slices allows anyone to decode $\mathbf{M}$** – and generates **random vectors** (RVs) $\mathbf{c}_{1..m,0}$ matching in size for each slice $\mathbf{m}$
2. Using **exclusive or** (XOR) operation between vector pairs of $\mathbf{m}_y$ and $\mathbf{c}_y$ S forms vectors $\mathbf{m}^{(0)}_{1..m}$, and by rearranging each $\mathbf{c}$ in a predictable fashion creates vectors $\mathbf{v}^{(0)}_{1..m}$
3. Through a mesh of $m \cdot n$ prearranged nodes $O_{1..m,1..n}$, all vector pairs of $\mathbf{m}^{(0)}$ and $\mathbf{v}^{(0)}$ are sent along m disjoint paths (referred to also as main path or *branch* from now on) in the form of **network messages** (NMs)
4. **Each intermediate node** $O_{y,x}$ of a given *stage* x (from now on also referred to as *column*) and branch y of the mesh:
    1. **receives input messages** $\mathbf{m}^{(x-1)}$, $\mathbf{v}^{(x-1)}$ and $\mathbf{c}_{1..m,x-1,y}$ from all the nodes in the previous stage $x-1$, except for the first stage nodes $O_{1..m,1}$, which receives only $\mathbf{m}^{(0)}$ and $\mathbf{v}^{(0)}$ from R

2. **generates** a RV $\mathbf{c}_{y,x}$ matching the received $\mathbf{m}^{(x-1)}$ and $\mathbf{v}^{(x-1)}$ in size
3. **creates outgoing messages** $\mathbf{m}^{(x)}$ (by applying $\mathbf{m}^{(x-1)}$ to $\mathbf{c}_{y,x}$ via XOR) and $\mathbf{v}^{(x)}$ (using XOR between $\mathbf{v}^{(x-1)}$ and the concatenation of $\mathbf{c}_{1..m,x-1,y}$)
4. **creates outgoing messages** $\mathbf{c}_{y,x,1..m}$ by slicing the locally generated $\mathbf{c}_{y,x}$ into m disjoint and equally sized pieces so that $\mathbf{c}_{y,x} = \|_{z=1}^{m} \mathbf{c}_{y,x,z}$
5. sends $\mathbf{m}^{(x)}$ and $\mathbf{v}^{(x)}$ **horizontally** and m pieces of $\mathbf{c}_{y,x,1..m}$ **diagonally**, except for the last stage ($x = n$) that sends all its outgoing messages to R

5. Finally R receives all the pieces sent by the stage n nodes and first recovers $\mathbf{m}_{1..m}$ by executing a sequence of XOR operations and then $\mathbf{M}$ (the RI) via the inverse operation of the original coding scheme

As visible, in this scheme:

- only nodes of the first stage know S
- and only the last stage knows R
- none of the nodes know both ($n \geq 2$)
- only R is able to do the decoding



S
R
$O_{1,1}$
$O_{2,1}$
$O_{m,1}$
$O_{1,2}$
$O_{2,2}$
$O_{m,2}$
$O_{1,n}$
$O_{2,n}$
$O_{m,n}$
$\mathbf{m}_1^{(0)},\mathbf{v}_1^{(0)}$
$\mathbf{m}_2^{(0)},\mathbf{v}_2^{(0)}$
$\mathbf{m}_m^{(0)},\mathbf{v}_m^{(0)}$
$\mathbf{m}_1^{(1)},\mathbf{v}_1^{(1)}$
$\mathbf{c}_{1,1,1}$
$\mathbf{c}_{1,1,2}$
$\mathbf{c}_{2,1,1}$
$\mathbf{c}_{2,1,m}$
$\mathbf{m}_2^{(1)},\mathbf{v}_2^{(1)}$
$\mathbf{c}_{2,1,2}$
$\mathbf{c}_{m,1,1}$
$\mathbf{c}_{1,1,m}$
$\mathbf{c}_{m,1,2}$
$\mathbf{m}_m^{(1)},\mathbf{v}_m^{(1)}$
$\mathbf{c}_{m,1,m}$
$\mathbf{m}_1^{(n)},\mathbf{v}_1^{(n)},\mathbf{c}_{1,n}$
$\mathbf{m}_2^{(n)},\mathbf{v}_2^{(n)},\mathbf{c}_{2,n}$
$\mathbf{m}_m^{(n)},\mathbf{v}_m^{(n)},\mathbf{c}_{m,n}$

Fig. 2: Message flows of the inner-layer

These for features together guarantee the **anonymity** and confidentiality of the inner-, while OR will provide the same for the outer-layer. Fig. 2 provides details on all the entities present in, and messages flowing within the inner layer.

Note: within this MT we consider a RV to be an **independent, equiprobable and memoriless binary string** (in other words “flow of bits”), so that knowing some part of it leaves the rest completely undetermined. In the Information Theoretical (IT) sense this means that the mutual information between any parts of such RV is 0.

5.3 Operations Within the Inner-layer

In this chapter we will demonstrate the very details of all the important operations done and entities present in the inner layer of the original solution. Fig. 3 serves the purpose of doing this with the freely chosen parameters of $m = 2$ and $m = 3$. Please note that in the original design [2] the demonstration of the mechanisms is done with the parameters $m = n = 3$, but – to provide ease of understanding and due to space constraints within the document – we present a smaller network. However it is important to emphasize that the scheme operates the same way, independently from these parameters (as long as $m, n \geq 2$).

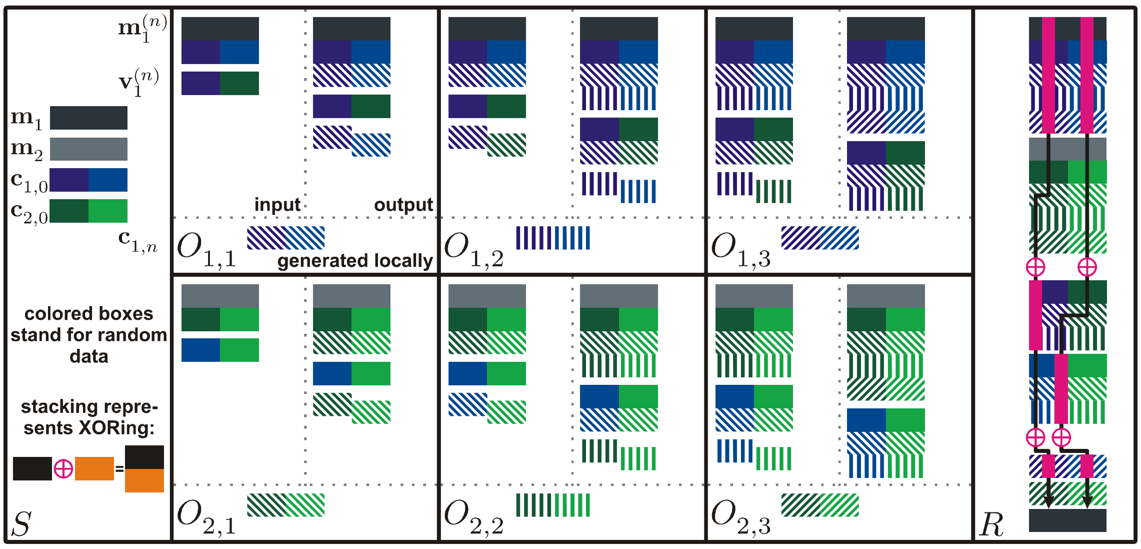


Fig. 3: Nodes' roles in the inner-layer

All the nodes what were represented by circles in Fig. 2 are now “zoomed” and become rectangular areas. Left and right sides represent sender S and receiver R (respectively), while the middle shows each relay node $O_{y,x}$ of the interjected mesh. Each relay node is divided into 3 sections, showing the messages the node **receives** (labelled by “input”), it **generates** by itself (“generated locally”) and – **after** executing **NC** – **sends** (“output”) to the next stage. Binary vectors are represented by boxes. Among these boxes those that are grey-scale stands for actual user data hiding $\mathbf{M}$, while coloured carets represent RVs. A given stacking of these boxes also represents a single binary vector, which is formed by “moulding” the original vectors together using (a series) XOR operation(s).

By looking at Fig. 2 and Fig. 3 one can see all the details of the coding and message flow that takes place in the mesh, but let us summarize it through the example of the node $O_{2,2}$. This node will **receive** the slice $\mathbf{m}_2^{(1)}$ {⬛🟩▨} of the secret (XORed by RVs) and the stacked RV $\mathbf{v}_2^{(1)}$ {🟦🟩} from $O_{2,1}$ (travelling via the same branch **horizontally**) and m pieces of $\mathbf{c}_{1..m,1,2}$ {▨, ▨} (travelling **diagonally**). At this moment

$O_{2,2}$ will **generate** its local RV $\mathbf{c}_{2,2}$ {▥}, **XOR** it to $\mathbf{m}_2^{(1)}$, forming $\mathbf{m}_2^{(2)}$ {⬛🟩▨▥}.

In addition it will also sequentially **concatenate** the received vectors $\mathbf{c}_{1..m,1,2}$ and **XOR** the resulting vector to $\mathbf{v}_2^{(1)}$, forming $\mathbf{v}_2^{(2)}$ {🟦🟩▨}. In the final coding step $O_{2,2}$ is to **slice** its locally generated RV $\mathbf{c}_{2,2}$ into m disjoint pieces, forming $\mathbf{c}_{2,2,1..m}$ {▥, ▥}.

The role of $O_{2,2}$ ends by **sending** $\mathbf{m}_2^{(2)}$, $\mathbf{v}_2^{(2)}$ and the m pieces of $\mathbf{c}_{2,2,1..m}$ to stage 3.

The duties of the first and the last stages are slightly simpler, as diagonal receiving and sending (respectively) does not take place.

The receiving node R will first decode each slice $\mathbf{m}$ through $2 \cdot m$ **XOR** operations done among the vectors indicated by Fig. 3 (XOR operation $\oplus$ between vector parts marked by ■). The XOR operations can take place in any order due to the following facts:

- XOR is **commutative** (as over the binary field it is equivalent to addition)
- $(\mathbf{b} \oplus \mathbf{a}) \oplus \mathbf{b} = \mathbf{a}$ for any binary vectors $\mathbf{a}$ and $\mathbf{b}$

After having the vectors $\mathbf{m}_{1..m}$ in hand, R can decode the original secret $\mathbf{M}$, which hides the **RI** used by the outer-layer, which altogether requires $2 \cdot m^2$ to XOR operations at R.

5.4 Churn Resilience and Failure Model

As deducible from the above explanations and visible from the proof in section III. of the paper [2] **decoding of RI** by any R is always possible as long as it **receives each an every message sent by the last stage** (n) unaltered. The pre-requisite of this is that all previous stages perform likewise. While error-free delivery of messages between stages can be guarantied by using connection oriented point-to-point services and/or integrity check and acknowledgment at the transport layer, these techniques **can not protect against message loss due to node failure**.

Herein we would like to define the failure model we will subsequently build upon: by **churn, we mean** the situation, when a **node fails to perform** any of its duties (receiving, RV generations, coding, forwarding):

- due to software or hardware failure
- because it leaves the P2P network in a controlled or uncontrolled fashion

- due to any network congestion or failure that prevents it to accept or make connections and to receive or send any/all of the messages due in the normal course of events

Because the system includes redundancy neither in the inner- nor in the outer-layer, **any failure will result in** the loss or corruption of at least one of the messages and – because of the fact that all messages are required for a successful decoding – in failure of some of the intended OR relays to acquire RI. As a direct consequence **the communication between S and D becomes completely impossible**.

Moreover both the inner- and the outer-layer communication are uni-directional, S will have no notion of the subsequent events and possible failures, therefore it can not take countermeasures in the form of trying to replace failed nodes or resending all retrying the communication entirely.

It is also important to emphasize that the number of nodes taking place in the communication is over $\ell \cdot m \cdot n$ and the messages traveling between those is $\ell \cdot m^2 \cdot n$.

As a conclusion, while offering advantages in the field of anonymity and computational complexity, the original system completely unprotected against node failures that are inherent to (unavoidable in) any P2P network.

6 Proposed Solution

Our proposed solution is based on the following two criteria: We will

- use (source) **coding techniques**
- rely on the **existing mechanisms** of the ES as much as possible

to **achieve resilience against churn**.

It is also obvious that in a typical OR system the number of real nodes is fixed to be low (typically $\ell = 3$), but in the ES the number of nodes necessary to reach these onion routers is quadratic, therefore the churn resilience has to target the inner-layer. If the high value of expected probability of node failure requires it, the sender S does have simple techniques (such as alternative OR path construction, hop-by-hop acknowledgment etc.) to to guaranty minimum reliability of the outer-layer as well. As the complexity of the problem lying in the inner-layer is considerably higher, we will focus on that and will not further discuss the details of the methods available for the outer-layer.

6.1 Naïve Approach

In this chapter we would like to present the most obvious solution to the problem and point out why – due to the design features of the ES – that would not be suitable to establish churn resilience. Such obvious approach is the following:

1. Increase the number of branches from m to $m' = m + r$, where r is chosen according to the probability of expected node failure or in other words the estimated or measured real-time reliability of the network
2. Use a coding scheme that can generate m' slices $\mathbf{m}'_{1..m'}$ out of $\mathbf{M}$ in a fashion that **receiving at least** m of those would **always allow decoding**, but receiving less then m **would never do so**
3. Select $m' \cdot n$ nodes for the inner-layer and send each $\mathbf{m}'$ along m' disjoint main paths (branches)

6.1.1 Threshold Secret Sharing Schemes

We need to find a **Secret Sharing Scheme** (SSS) that satisfies the requirements of point 2. in chapter 6.1. The first part of the statement ("receiving at least") comes from the fact that we proportioned our scheme to tolerate r node failures, therefore we expect the new scheme to always be able to decode, as long as the number of node failures is at most r. The second part of our requirements ("receiving less than") is a constraint that we must explicitly inherit from the ES.

Coding schemes satisfying this very requirements do exist and form the category of **Threshold Secret Sharing Schemes** (TSSS). One of the most notable of the actual construction techniques belonging to this family is **Shamir's Secret Scheme** (SSSS) [1]. Its features can be summarized by the following sentence: it allows arbitrary secret $\mathcal{D}$ be encoded by $\mathcal{N}$ shares in a way that receiving any $\mathcal{K}$ ($\mathcal{N} \geq \mathcal{K} \geq 1$) of those completely determines $\mathcal{D}$, while receiving less than $\mathcal{K}$ leaves $\mathcal{D}$ completely undetermined. In the IT interpretation we can rephrase this by saying that in case of such a $(\mathcal{K}, \mathcal{N})$ TSSS when one receives $\mathcal{M}$ shares only the following two possibilities exist:

- iff $\mathcal{M} \geq \mathcal{N}$ then $H(\mathcal{D}) = 0$, where H stands for **entropy**
- otherwise $H(\mathcal{D}) = H(\text{source})$

In such scheme we call $\mathcal{K}$ the **threshold** and it satisfies the criteria of **perfect**ness.

It is not stated explicitly by the ES [2], but by their requirements against the coding technique that produces $\mathbf{m}_{1..m}$ vectors from $\mathbf{M}$, we make the assumption that they build on an unspecified (m, m) TSSS. Construction of such a scheme is very simple. An example may be the following:

1. Generate $m - 1$ RV of equal size (that of $\mathbf{M}$), forming $\mathbf{m}_{1..m-1}$
2. XOR each of these to $\mathbf{M}$, forming $\mathbf{m}_m$

Such construction is **computationally lightweight** (from the perspective of both, the coding and decoding steps) and its **necessary and sufficient conditions are the reception of all m slices**.

For the cases when $\mathcal{K} < \mathcal{N}$, the construction is much less obvious and has drawn considerable research attention. Currently existing solution range from polynomial oversampling (proposed by Shamir himself and requiring expensive operations, such as **Lagrange interpolation** [1]) to some schemes of late relying only on computationally lightweight operations (such as XOR [8][12]).

As more detailed discussion of the SSS is outside of the scope of this MT, therefore we will purely rely on the assumptions that such scheme exists and is computationally feasible. For the sake of later analysis, we emphasize that TSSS is a **perfect** (which implies $|\mathbf{m}| \geq |\mathbf{M}|$) SSS and we assume it to be **ideal** (which means $|\mathbf{m}| \leq |\mathbf{M}|$) resulting in the size of each share being equal to that of the secret ($|\mathbf{m}| = |\mathbf{M}|$).

6.1.2 Single-redundant Coding

Let us discuss the details of how the coding explained in chapter 6.1.1 would operate and point out its weakness, that calls for additional measures. Fig. 4 compares the slicing technique utilized by the ES and the method described under 6.1.1 (upper) in case of the parameters of $\mathcal{K} = 2$ and $\mathcal{N} = 3$ (lower). From now on we will not use a different notation for $\mathbf{m}'$ and $\mathbf{m}$ as those are in fact same with regards to their size ($|\mathbf{m}'| = |\mathbf{m}| = |\mathbf{M}|$) and content: slices that encode $\mathbf{M}$, totally hide/give full access to its original content when certain conditions are not met/not met (respectively).

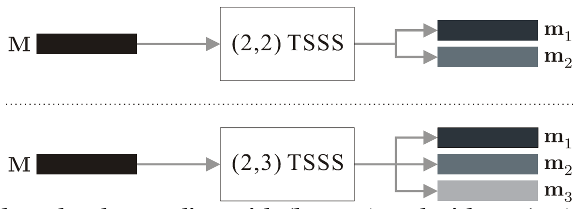


Fig. 4: Single-redundant coding with (bottom) and without (top) redundancy

From the network's structure's perspective, applying the extended scheme (2,3) TSSS will do nothing more than creating an **additional branch** in the network, in a way that if R receives and can decode any 2 of the 3 slices ($\mathbf{m}_1$ and $\mathbf{m}_2$ or $\mathbf{m}_2$ and $\mathbf{m}_3$ or $\mathbf{m}_1$ and $\mathbf{m}_3$), then he can also recover $\mathbf{M}$, while receiving less than 2 ($\mathbf{m}_1$ or $\mathbf{m}_2$ or $\mathbf{m}_3$ or none of them) will leave him in the same situation as in case of the ES. Needless to say that an attacker – trying to compromise the network – has the same objective, which is trying to receive least 2 of the 3 slices.

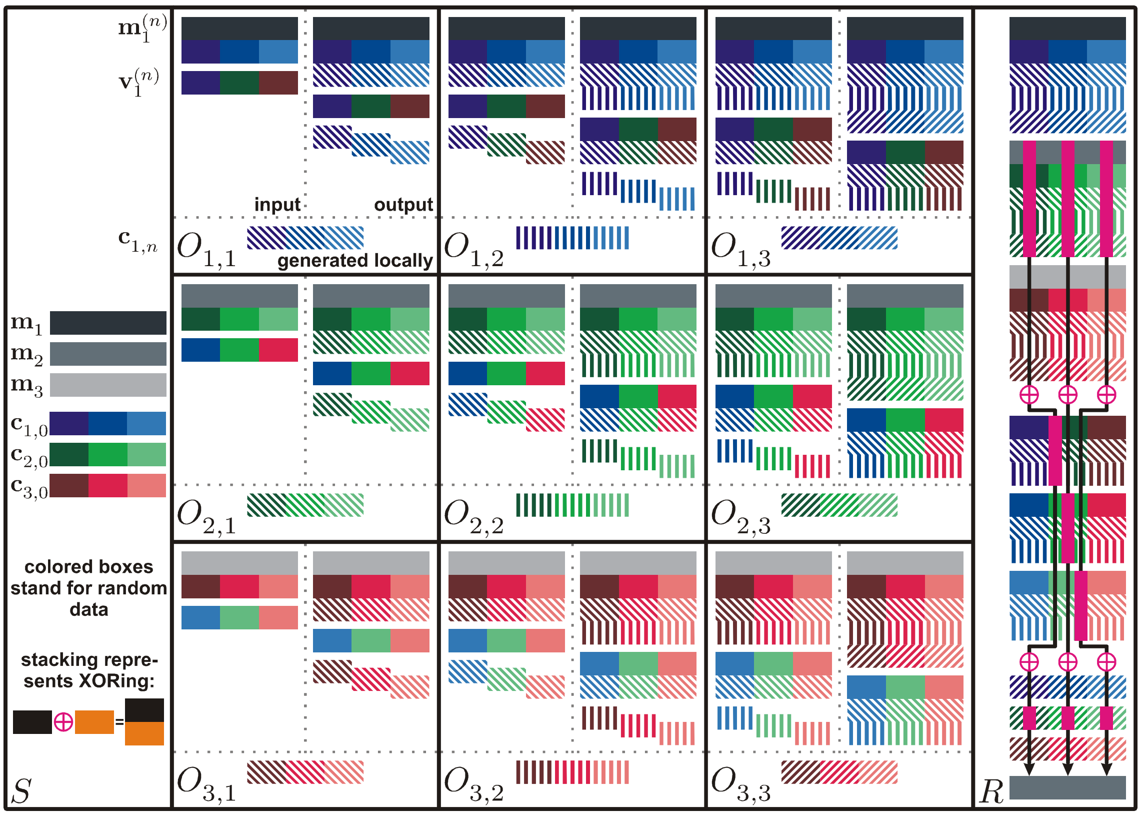


Fig. 5: Single-redundancy: decoding

Fig. 5 demonstrates how such an extension of the ES would look in practice, including highlighting the decoding steps for $\mathbf{m}_2$. As visible the scheme works as before, except for the number of pieces the generated RVs are cut into.

Now – through an example – let us see how the system reacts to churn. Fig. 6 shows the case when node $O_{1,3}$ fails to meet its duties. Let us linger a bit around the details of the figure. The grey area represents the node that fails, including the case when any of its messages fails to get delivered unaltered. Messages (vectors) covered by black carets are the ones that are lost in the worst case, as a direct result of node failure (churn). What **we expect** is that **decoding** would still **be possible**, as R:

- did use (2,3) TSSS to create each $\mathbf{m}$ vector
- and it did receive 2 of those ($\mathbf{m}_2$ and $\mathbf{m}_3$) unaltered

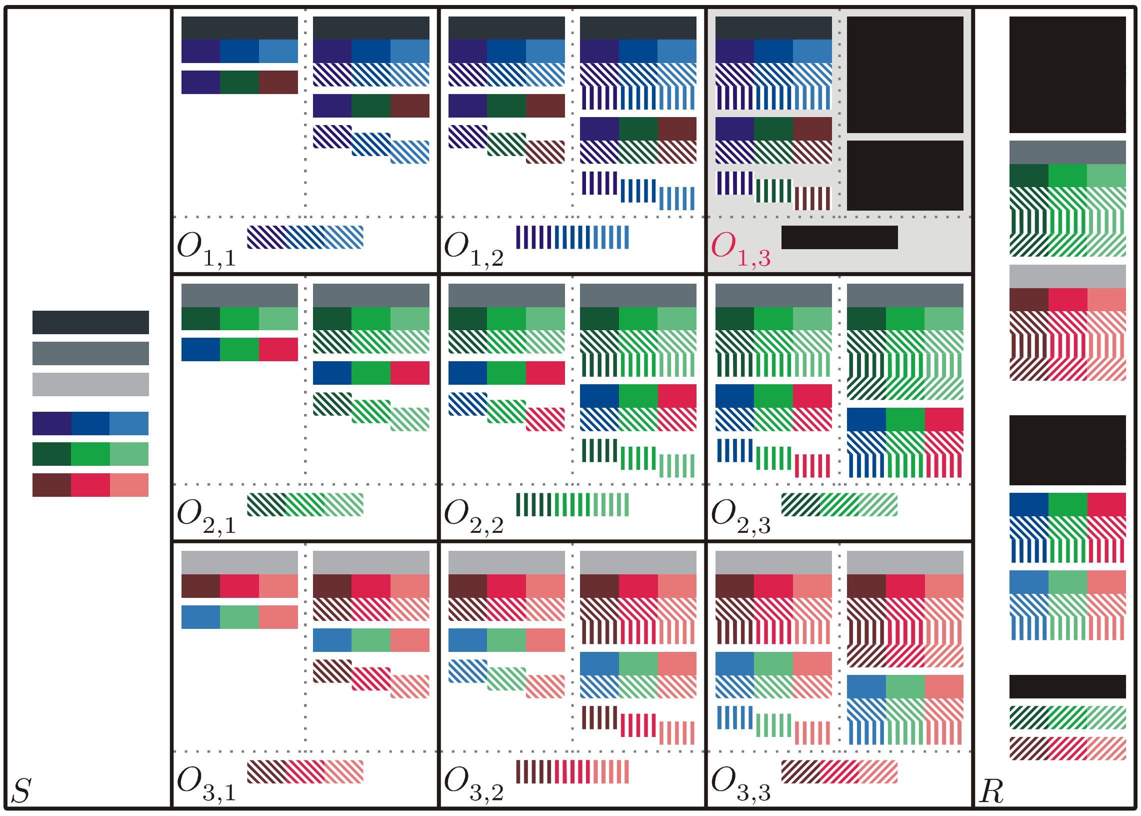


Fig. 6: Single-redundancy: failure in branch one

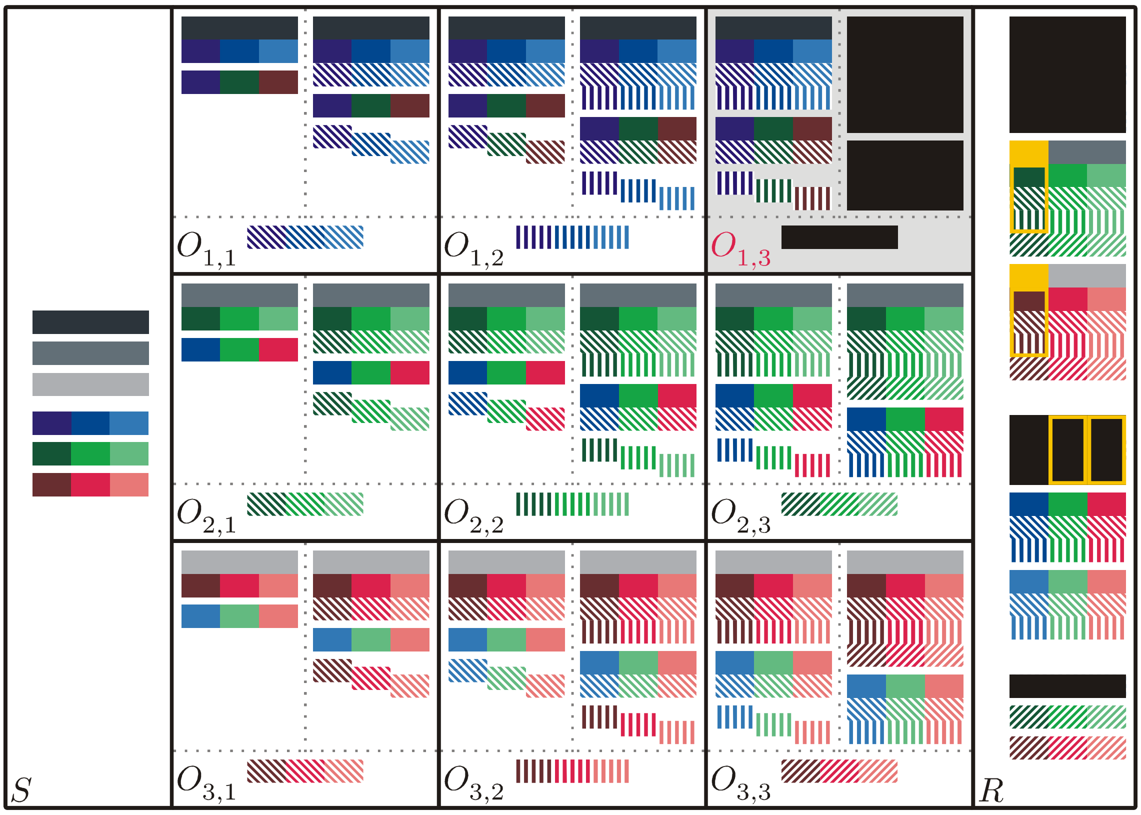


Fig. 7: Single-redundancy: vectors affected by churn (branch 1)

Unfortunately the above mentioned assumption is incorrect, due to the special nature of the ES. More precisely, due to the fact that RVs used during the encoding – and therefore necessary for the decoding – of an $\mathbf{m}_y$ travelled diagonally and accumulated in **all** m **branches** of the network. Fig. 7 points out those RVs that become lost due to the failure of $O_{1,3}$, that would be necessary for decoding the slices travelling via otherwise unaffected branched ($\mathbf{m}_2$ and $\mathbf{m}_3$). These RVs are marked by yellow carets {▭}, while the affected parts of the slices are highlighted by yellow rectangles {🟨}.

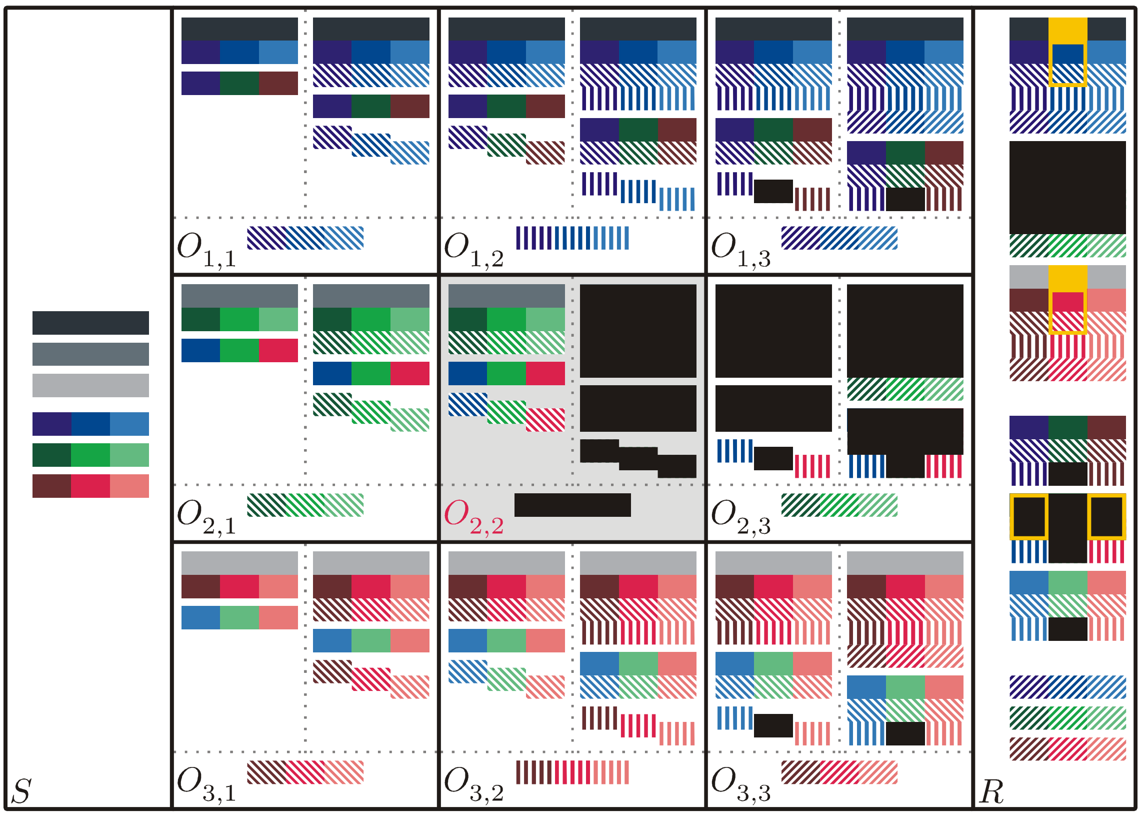


Fig. 8: Single-redundancy: vectors affected by churn (branch 2)

Quick summary: as visible, while R did receive 2 slices, it will be unable to decode those, due to the fact that part of the RVs necessary to do so are also lost.

Using the same notations, et us quickly demonstrate the cases when failure occurs in other branches (and stages). Fig. 8 and Fig. 9 show the cases when nodes $O_{2,2}$ and $O_{3,1}$ (respectively). Fig. 5 is available for reference for the invisible vectors.

Conclusion: as visible through the demonstrated examples, a failure in branch y, will cause not only the loss of vector $\mathbf{m}_y$, but also decoding failure in the y^{th} part **all other slices** travelling through the remaining branches.

Because of this phenomenon inherent to the design, a single-redundant coding would not be sufficient to counter node failure (churn).

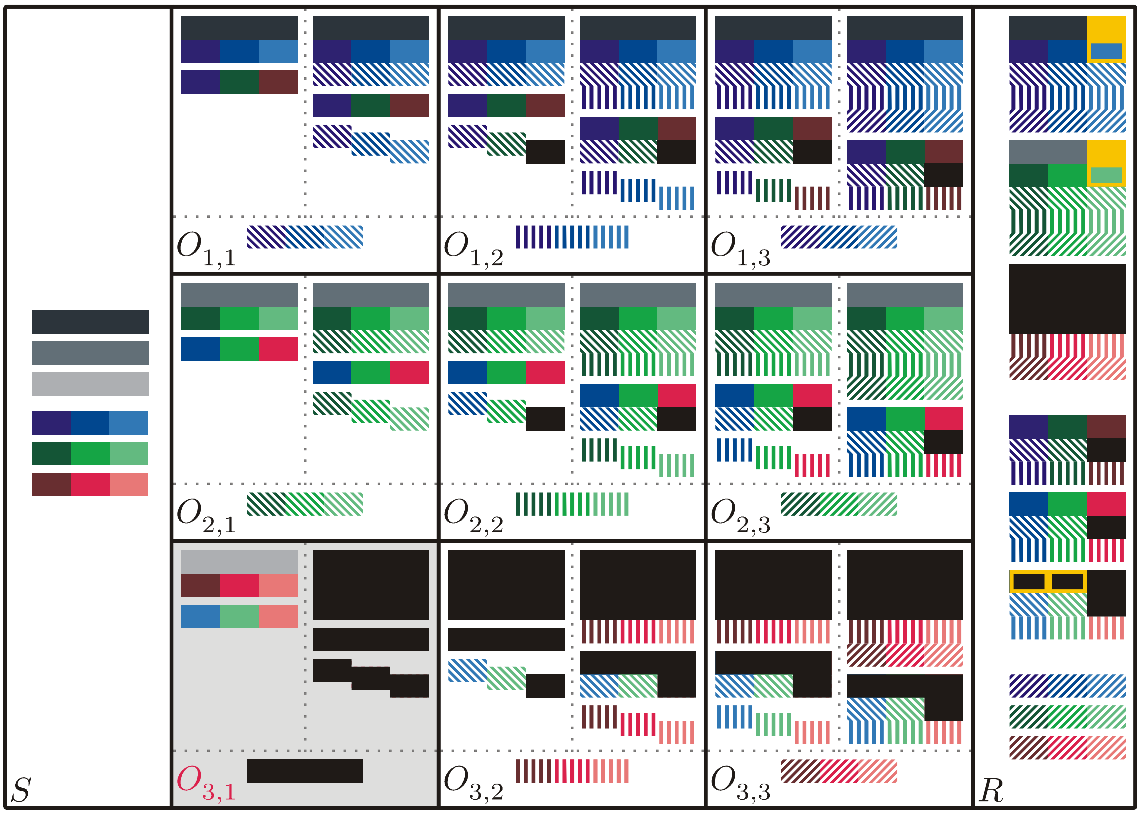


Fig. 9: Single-redundancy: vectors affected by churn (branch 3)

6.2 Double-redundant Coding

6.2.1 Encoding

Our final proposal builds on the naïve approach, while stepping beyond it. The feature that we take advantage is the following phenomenon deducible from the construction of the ES: in case of failure of c relay nodes, at most c disjoint parts (each $\frac{|\mathbf{m}|}{m}$ in size) of all the $\mathbf{m}$ vectors become un-decodable via XOR in the remaining branches.

Based on this observation we introduce redundancy also to the slices already encoded using TSSS by re-applying the same scheme again, with the same parameters. Fig. 10 shows how this is achieved in case of $\mathcal{K} = 2, \mathcal{N} = 3$.

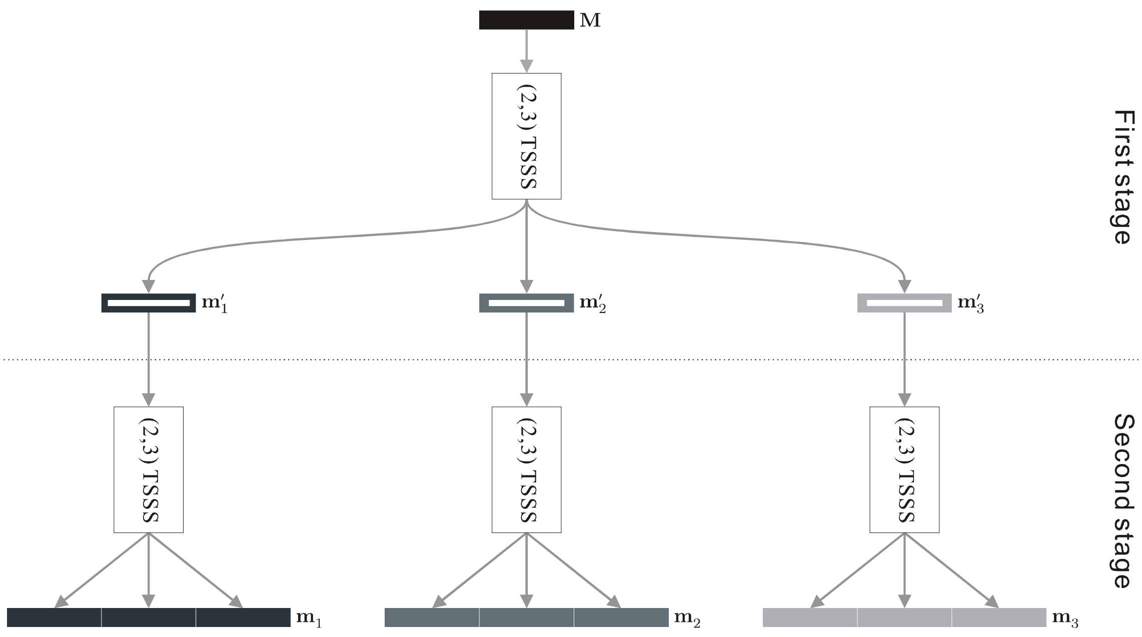


Fig. 10: Proposed solution using double-redundancy

The operation necessary to achieve properly encoded slices is as follows:

1. Based on the ES (that provides the parameters m and n, as shown in Fig. 2 and Fig. 3), **choose the level of churn resilience** required under the present conditions. Let us call this value r (stands for redundancy)
2. **Encode** $\mathbf{M}$ using (m_{ES}, m) TSSS (where $m = m_{\text{ES}} + r$) into m intermediate slices $\mathbf{m}'_{1..m}$ (let us refer to this as **first stage coding**).
3. **Encode each** $\mathbf{m}'$ into m subsequent slices
4. **Concatenate** the resulting vectors that were generated from the same intermediate slice $\mathbf{m}'_y$, resulting in $\mathbf{m}_{1..m}$ vectors (let's use the tern **second stage coding** for these last two steps)
5. The rest goes according to the technique presented in chapter 5.2, bearing in mind the updated network parameters ($m = m_{\text{ES}} + r$ and $n = n_{\text{ES}}$).

We would like to point out that – using this scheme – the second stage will cause **linear size increment** of all the $\mathbf{m}$ vectors, compared to the ES. Because of this, all relevant messages ($\mathbf{m}_{1..m}^{(0..n)}$, $\mathbf{v}_{1..m}^{(0..n)}$, $\mathbf{c}_{1..m,1..n,1..m}$ and $\mathbf{c}_{1..m,0}$) will also increase in size according to the same multiplier.

Important notes: For the case of simplicity and to conserve space, from this point on we will refer to

- figures Fig. 6 - Fig. 9 in the context of the **Proposed Scheme** (PS) described herein (chapter 6.2). This is possible, because from that perspective the PS looks and – in between the encoding and decoding steps (exclusive) – acts like single-redundant solution, except for the size increment mentioned in the previous paragraph. In other words the nodes of the mesh or not aware of the details of the coding scheme utilized by S
- m_{ES} as k. In other words it also holds that $k + r = m$, which is analogous to the naming used in conjunction with the TSSSs

6.2.2 Decoding in General

Upon receiving the messages, R will be able to acquire $\mathbf{M}$ as long as the number of node failures does not exceed r using the following algorithm, including the execution of the inverse of the encoding operations in **reverse order**:

1. Locate node any possible node failure by looking at all the received vectors from among $\mathbf{m}_{1..m}^{(n)}$ and $\mathbf{v}_{1..m}^{(n)}$ and locate the node failure(s) in the form of y_1, y_2 … y_d (where y is the branch-id where failure occurred)
2. Make sure that $d \leq r$
3. **Discard** the $y_1\mathrm{th}$, $y_2\mathrm{th}$ … $y_d\mathrm{th}$ $\frac{1}{m}\mathrm{th}$ of each received $\mathbf{m}^{(n)}$ and $\mathbf{v}^{(n)}$ vectors
4. Decode the remaining parts of the received $\mathbf{m}^{(n)}$ and $\mathbf{v}^{(n)}$ vectors via the usual means (by applying XOR operation)
5. Apply the **inverse operation** (decoding step) of (m_{ES}, m) TSSS used at **second stage** of the encoding step to obtain any k pieces of $\mathbf{m}'$ vectors
6. Feed these to the **inverse operation** of (k, m) TSSS used at **first stage** of the encoding step to obtain vector $\mathbf{M}$ (RI)

6.2.3 Decoding Example

Let us make a close demonstration of the decoding scheme through the example presented in Fig. 8. As visible, in that situation $O_{2,2}$ is affected by churn, therefore vectors $\mathbf{m}_2^{(3)}$ and $\mathbf{v}_2^{(3)}$ will not be received. Moreover the 2nd 1/3rd of $\mathbf{m}_1^{(3)}$ and $\mathbf{v}_3^{(3)}$ become impossible to decode. The decoding of $\mathbf{M}$ in this situation will be as shown and explained herein:

1. Identify the failed branch and discard relevant parts of otherwise received vectors, marked by red crosses {✗}: **left side**
2. Carry out the XOR operations and execute the inverse operation of the second stage (2, 3) TSSS: **right side**
3. Using the resulting temporary vectors, execute the inverse of the first stage (2, 3) TSSS, providing $\mathbf{M}$: **bottom**

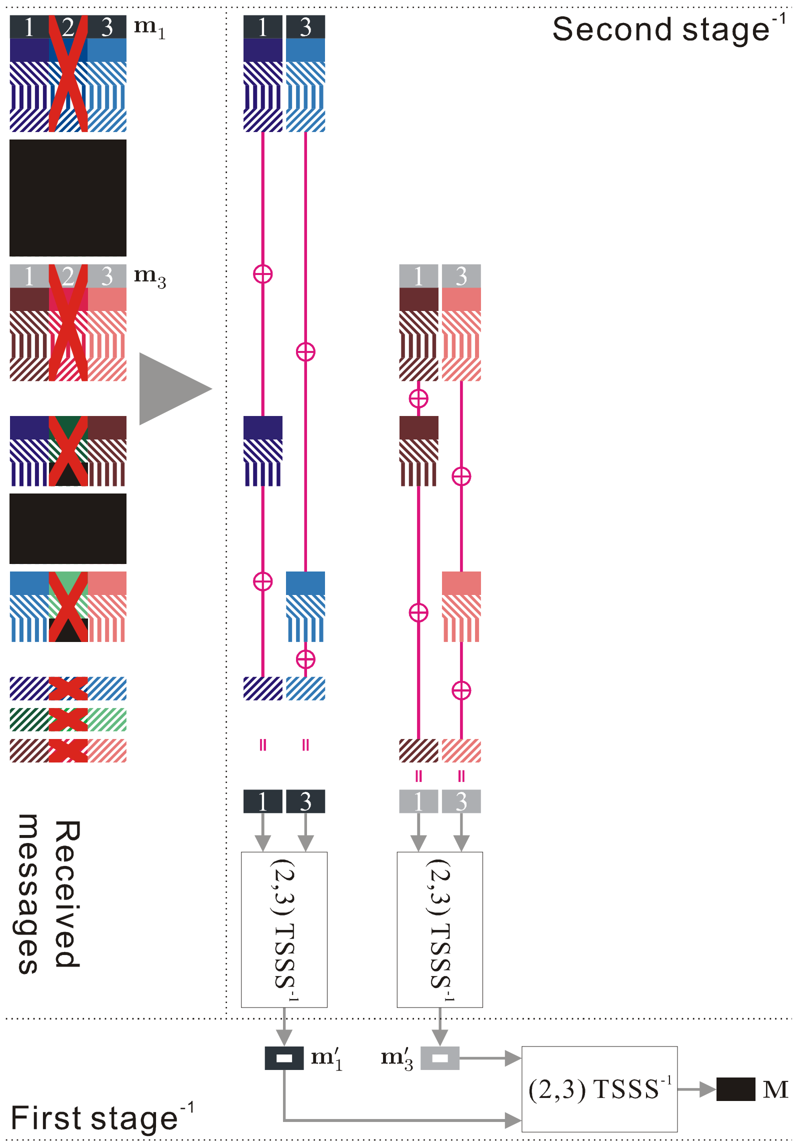


Fig. 11: Double-redundancy: decoding

6.2.4 An Interesting Phenomenon

Careful reading of the decoding algorithm explained in chapter 6.2.2 reveals an interesting – and advantageous – property of this scheme which can be phrased the following way: a failure of a node $O_{y,x}$ will render yth $\frac{1}{m}$th of all the received $\mathbf{m}^{(n)}$ and $\mathbf{v}^{(n)}$ vectors unusable (undecodable via XOR), therefore **any other node failure within the same branch y will not further degrade the decoding performance**. This can be summarized into a theorem describing the properties of our scheme.

Theorem: Message M can be recovered as long as the number of **branches where any churn is present is at most** r.

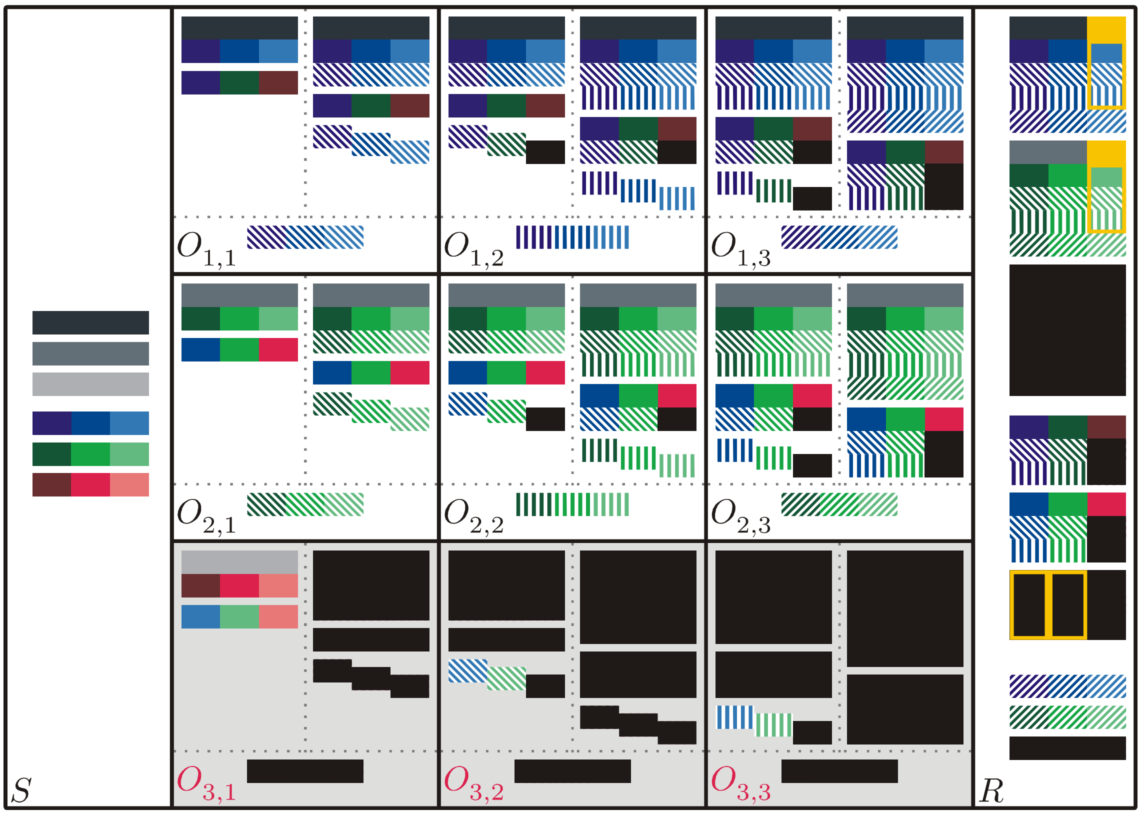


Fig. 12: Double-redundancy: recovery from branch failure

Based on our earlier examples Fig. 12 demonstrates a case when a whole branch fails ($O_{3,1}$, $O_{3,2}$ and $O_{3,3}$), decoding is still possible using the algorithm described under chapter 6.2.2, simply because additional node failure in the same branch affects vectors that – due to earlier failures – are either already corrupted or anyway would not be necessary for (would be discarded by) the decoding process.

6.2.5 Non-atomic Node Failure

In chapter 5.4 we assumed churn to be an atomic event, in the sense that either all messages of a node are lost or none of them are. The decoding scheme has also built on this assumption. While it is true that this is the simplest case, it is easy to extend the PS to partial node failure.

Such failure may occur when a node is fully functional and performs its tasks correctly, but – for example due to limited network connectivity – it fails to reach some (even one) of its correspondents. Please recall the discussion of chapter 5.4 that said a loss of any of the transmitted vectors will result in an utter failure of decoding of RI.

Limited – or selective lack of – connectivity are known phenomena and should be addressed. Both of the following two methods shall be sufficient to address this scenario:

1. Encode presence information into all the $\mathbf{m}^{(x)}$ or $\mathbf{v}^{(x)}$ vectors, indicating which of the random vectors $\mathbf{c}_{1..m,1..n,1..m}$ have been received and used during the encoding. Using this technique, R can identify the failed branches (please see chapter 6.2.4 for more details on this) by enumerating the source O nodes that – under normal circumstances – would have sent the missing pieces. For example if everything has been received and used for encoding, except for the pieces:
   - Failures in the same branch: If $\mathbf{c}_{2,2,1}$ and $\mathbf{c}_{2,2,3}$ are missing, then make sure that $r \geq 1$ and proceed with the decoding while considering branch 2 to have failed
   - Failures in different branches: If $\mathbf{c}_{2,2,1}$ and $\mathbf{c}_{3,2,1}$ are missing, then make sure that $r \geq 2$ and decoding under the assumption that branches 2 and 3 have failed
2. Handle such scenario **as if it was tampering** and rely on the Tampering Resilience (TR) technique discussed in chapter 8

Our recommendation is to follow the second approach, for at least the following two reasons:

- TR will anyway be present, therefore can be taken advantage of without additional design or implementation efforts
- Including additional information into the vectors also increases attacker's practical chances to compromise anonymity. Under normal conditions, the system can be constructed so that attackers have to execute very large number of oper-

ations to be able to decode RI (see details on this in chapter 7.1.3). If a malicious node O can make an educated guess on its location in the mesh, then the attackers can rule out a large amount of invalid decoding combinations, this way weakening the level of practical anonymity.

6.3 The Inner-layer processes Formally

To support ease of reproducibility, this chapter will give a formal definition of all the inner-layer mechanisms described in the earlier chapters. Please also refer to Fig. 2. Additional information can be found in the related papers [2] [5].

6.3.1 Encoding

As a preparation step, S performs two times threshold secret sharing on the message $\mathbf{M}$ to be sent as follows:

1. S shares the message $\mathbf{M}$ into m intermediate shares $\mathbf{m}'_{1..m}$ using a $(m-r,m)$ TSSS. If $\mathbf{M} \in \mathrm{GF}(2)^a$ then $\mathbf{m}'_{1..m} \in \mathrm{GF}(2)^a$
2. S shares each of the intermediate shares $\mathbf{m}'_{1..m}$ via the same TSSS into m subsequent shares, each, producing $\mathbf{m}'^{1..m}_{1..m}$ ($\mathbf{m}'^{1..m}_{y}$ are the m shares produced from $\mathbf{m}'_y$)
3. S produces a message $\mathbf{m}_y = ||_{z=1}^{m} \mathbf{m}'^{z}_{y}$ for $\forall y \in [1,m]$, where $\mathbf{m}_{1..m} \in \mathrm{GF}(2)^d$ (where $d = am$)

In addition S forms vectors $\mathbf{m}^{(0)}_{1..m}$ the following way: $\mathbf{m}^{(0)}_y = \mathbf{m}_y \oplus \mathbf{c}_{y,0}$ for $\forall y \in [1,m]$, where $\mathbf{c}_{1..m,0}$ are m random vectors and $\mathbf{c}_{1..m,0} \in \mathrm{GF}(2)^d$. Subsequently S divides every $\mathbf{c}_{y,0}$ into m equally sized consecutive pieces called $\mathbf{c}_{y,0,1..m}$.

Let $\mathbf{V}_S \in \mathrm{GF}(2)^{d\times d}$ be the following coefficient matrix with $m \times m$ elements:

$$\mathbf{V}_S = \begin{bmatrix} \mathbf{c}_{1,0} \\ \mathbf{c}_{2,0} \\ \vdots \\ \mathbf{c}_{m,0} \end{bmatrix} = \begin{bmatrix} (||_{z=1}^{m} \mathbf{c}_{1,0,z}) \\ (||_{z=1}^{m} \mathbf{c}_{2,0,z}) \\ \vdots \\ (||_{z=1}^{m} \mathbf{c}_{m,0,z}) \end{bmatrix}$$

In addition let $\mathbf{v}_{1..m}^{(0)}$ be formed the following way: $\mathbf{v}_y^{(0)} = \|_{z=1}^{m} \mathbf{c}_{z,0,y}$ for $\forall y \in [1, m]$.

Now these messages $\mathbf{m}_{1..m}$ and $\mathbf{v}_{1..m}$ are ready to be transmitted over the m branches of the inner layer as shown by Fig. 2.

6.3.2 Forwarding and Network Coding

Each intermediate node $O_{y,x}$ (for $\forall x \in [1, n]$ and $\forall y \in [1, m]$) of the forwarding mesh will proceed the following way:

1. Choose random coding coefficient $\mathbf{c}_{y,x} \in \mathrm{GF}(2)^d$ to encode the input message $\mathbf{m}_y^{(x-1)}$ the following way: $\mathbf{m}_y^{(x)} \oplus \mathbf{c}_{y,x}$
2. Calculate $\mathbf{v}_y^{(x)}$ using the input message $\mathbf{v}_y^{(x-1)}$ as follows:
    1. For $x = 1$: $\mathbf{v}_y^{(1)} = \mathbf{v}_y^{(0)}$
    2. Otherwise: $\mathbf{v}_y^{(1)} = \mathbf{v}_y^{(x-1)} \oplus (\|_{z=1}^{m} \mathbf{c}_{z,x-1,y})$
3. For $x < n$, m output messages $\mathbf{c}_{y,x,1..m}$ are created by slicing $\mathbf{c}_{y,x}$ into m equally sized consecutive parts so that $\mathbf{c}_{y,x} = \|_{z=1}^{m} \mathbf{c}_{y,x,z}$
4. Finally:
    1. For $x < n$, messages $\mathbf{m}_y^{(x)}$ and $\mathbf{v}_y^{(x)}$ are forwarded to $O_{y,x+1}$, while each $\mathbf{c}_{y,x,z}$ is sent to $O_{z,x+1}$
    2. For the nodes of the last stage ($x = n$) all messages $\mathbf{m}_y^{(n)}$, $\mathbf{v}_y^{(n)}$ and $\mathbf{c}_{y,n}$ are sent directly to R itself

6.3.3 Decoding

As R receives all the parts that have taken part in the NC process, the decoding is always possible through the following steps:

1. R can construct the following matrix:

$$\mathbf{V}^{(n)} = \begin{bmatrix} \mathbf{v}_1^{(n)} \\ \mathbf{v}_2^{(n)} \\ \vdots \\ \mathbf{v}_m^{(n)} \end{bmatrix} = \begin{bmatrix} \|_{z=1}^{m} \{\oplus_{s=0}^{n-1}\mathbf{c}_{z,s,1}\} \\ \|_{z=1}^{m} \{\oplus_{s=0}^{n-1}\mathbf{c}_{z,s,2}\} \\ \vdots \\ \|_{z=1}^{m} \{\oplus_{s=0}^{n-1}\mathbf{c}_{z,s,m}\} \end{bmatrix}$$

2. Each $\mathbf{m}_y$ can be computed the following way:

$$\mathbf{m}_y = \mathbf{m}_y^{(n)} \oplus \mathbf{c}_{y,n} \oplus \left(\|_{z=1}^{m} \{\oplus_{s=0}^{n-1}\mathbf{c}_{y,s,z}\}\right) \text{ for } \forall y \in [1, m]$$

3. The recovery of $\mathbf{M}$ is directly possible via the decoding of the selected $(m-r, m)$ TSSS using any $m-r$ of the $\mathbf{m}_{1..m}$ vectors

6.4 The Coding Scheme of the Second Stage

Until now, we always suggested that both, the first- and the second-stage, coding schemes use the same type of function: a TSSS. In fact this is not completely mandatory and we will shows that, if done otherwise, there can be improvements of some properties of the PS.

The main motivating factor behind choosing TSSS has been explained under chapter 6.1.1 in details. As visible, the only reason why TSSS has been chosen for the first-stage is requirement inherited from the ES, but such requirement has not been phrased against the properties of the $\mathbf{m}_{1..m}$ vectors. In practice, this means that we have the freedom of replacing the coding scheme used by the second-stage, as long as the new code holds the following property: **the chosen coding scheme must be able to** successfully **decode each** $\mathbf{m}$ as long as **at most** $\frac{r}{m}\mathrm{th}$ of those become un-decodable via XOR due to loss of relevant RVs**.**

During our research we have evaluated multiple coding schemes to find out which suits our requirements the best. One branch of this research has aimed at using Fountain Codes as second-stage code – as for example Raptor Codes offer advantageous properties, such as linear time complexity – but we found that the following property of those make them inappropriate for our case: even receiving m slices would not provide $p = 1$, where p stands for the probability of successful decoding.

Finally we settled on the family of erasure codes, that would provide exactly what we are looking for:

1. **Guaranteed decodability** as soon as the fixed threshold of received fraction of the original message arrives
2. **No assumption on** the way **entropy** behaves while under this threshold

The answer to the reasonable question that asks why we would further research a topic that already has a reasonable solution (discussed in chapter 6.2.1) lies in the realms of **efficiency and network economy**.

6.4.1 Size Expansion, Network Economy

As explained earlier, recursively applying a TSSS will cause size-expansion that is **quadratic** for each inner-layer. While it is true that the subsequent coding schemes may rely on computationally very efficient operations (XOR), in some environments the size of the messages travelling over the network might still be an issue, mainly because their number lies in $\mathcal{O}(n^3)$ or in $\mathcal{O}(n^4)$ (depending on whether we consider ℓ to be constant or not).

Due to the constraints explained above, the only place where we can achieve more relaxed growth is the second-stage code. If we use something more economic than

TSSS as second-stage code, then we can achieve **sub-quadratic** (our target is linear) size-growth.

Let us turn our focus to the **second stage** now. It is easy to see that under no condition will we able to find a scheme that – for every possible input – can achieve a growth rate better (lower) than $R = \frac{\tilde{k} + \tilde{r}}{\tilde{k}}$ (where $\tilde{k}$ is size of the encoder's input and $\tilde{r}$ stands for the size of redundancy added by the encoder). This is because if were to achieve an expansion rate lower than this, that would imply that $|\mathrm{decode}(\mathbf{d})| > |\mathbf{d}|$ to be true for every possible $\mathbf{d}$ which is a contradiction (following the **pigeonhole principle** and the fact that there must a **bijection between the inputs and the outputs** of the coding scheme).

Fig. 13 shows the amount of data that travels through the network in case TSSS is applied twice, instead of using a – yet imaginary, but – optimal scheme. The graph values represent the mount of bytes that needed to be sent, under the assumption that RI consists of 1 byte and $r = 2$.

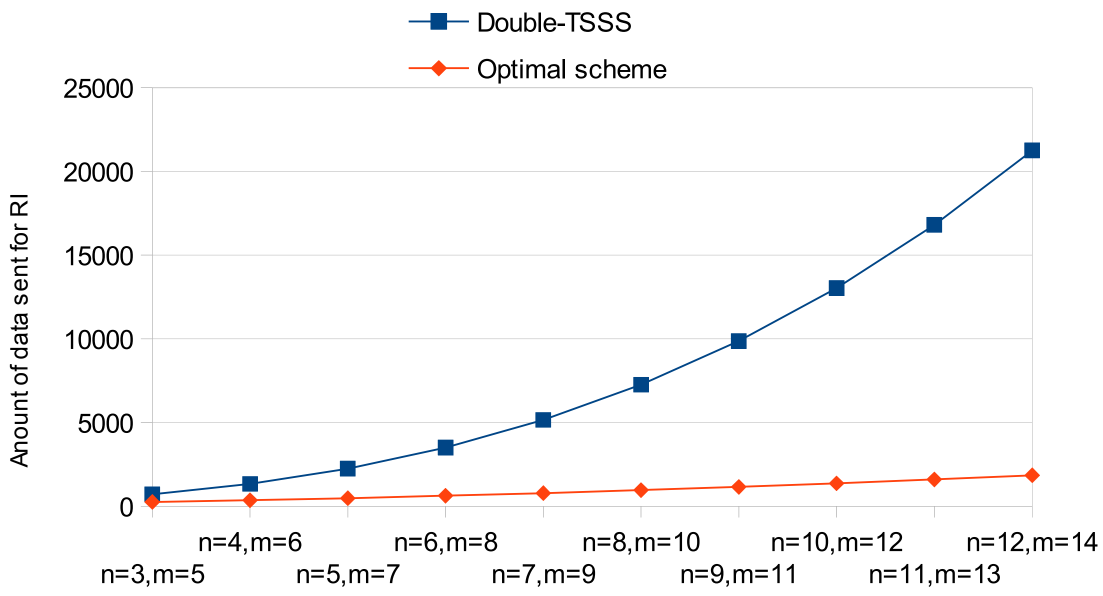


Fig. 13: Proportional amount of data sent for RI (r=2)

The actual format of the RI has not been defined by the original design [2], but **for** the example it was **100 bytes**, **double-TSSS** solution would send over **20 Mbytes**, while an **optimal approach** would manage to do the same with under **2 Mbytes** in a network with the parameters $m = 12, n = 14$.

6.4.2 Reed-Solomon Code

In chapter 6.4.1 we have shown that the best error correcting scheme we can find may provide the expansion rate of $R = \frac{\widetilde{k} + \widetilde{r}}{\widetilde{k}}$, so in this chapter we attempt to find a scheme as close to this limit as possible. In addition to trying to solve this problem, we will also propose to use a specific type of code construction to support ease on the decoder's side. In this chapter we would like to demonstrate a practical solution for using an EC that can provide the advantages described in chapter 6.4.1. The actual EC we choose is a special class of the **Error Correcting Codes** (ECC) called **Reed-Solomon Code** (RSC).

RSC is a **Maximum Distance Separable** (MDS) **linear code**, which – in other words – means that – during the encoding – the addition of each redundant symbol successfully **increases the code distance**. An $(\widetilde{n}, \widetilde{k})$ RSC transmits $\widetilde{k}$ source symbols in the form of $\widetilde{n}$ channel symbols. As it is a Maximum Distance Separable Code (MDSC) it has the capability of correcting $\lfloor (\widetilde{r} + 1)/2 \rfloor$ symbol changes and $\widetilde{r}$ symbol **erasures**, where $\widetilde{r} = \widetilde{n} - \widetilde{k}$. **Our interest lies in the code's erasure recovery capabilities**.

An additional property we can take advantage of is the **systematic construction** of some RSC. By systematic we mean the fact that the output of the encoder will include its input at a specific location. For example if we use a **prefix code** the vector $\mathbf{m}$ will take the form of $\mathbf{m}' \parallel \mathbf{r}$, where $\mathbf{r}$ stands for the vector of redundant symbols created by the encoder. It is not absolutely necessary for the code to be systematic, but makes the decoder's life much easier and more efficient in the general case, as – if the system has not been burdened by node failures, then – the decoding consists of only stripping off of $\mathbf{r}$.

Let us look at an example of how RSC may be used to provide the required erasure resilience through the popular $(255, 223)$ systematic RSC ($\widetilde{s} = 8$ thus symbols are bytes) also used by NASA that also has FPGA-based implementations. According to the earlier explanation, this code will be able to encode a 223-byte long $\mathbf{m}'$ vector into a 255 byte long $\mathbf{m}$ vector in a way that – if the location of the erasures is known to the receiver – decoding of $\mathbf{m}'$ will be possible as long as the number of bytes lost does not exceed $\widehat{r} = 32$. As $\mathbf{m}'$ stands for the vector that is the output of the first stage encoder (TSSS) and the input of its decoder, recovering $\mathbf{m}'$ is the condition of being able to recover the RI represented by $\mathbf{M}$.

We need to exploit one more technique often used with this code construction, to be able to directly use this $(255, 223)$ to provide arbitrary erasure-resilience properties. This property is called **code shortening** and it refers to the following idea: in case the number of symbols to be encoded $i = |\mathbf{d}|$ is less than $\widetilde{k}$ (223 in this case) it is possible to proceed the following way to be able to be able to take advantage of the chosen coding scheme:

1. Pad $\mathbf{m}'$ with $\widetilde{k} - i$ **zeroes, resulting in** $\widetilde{\mathbf{m}}'$ being $\widetilde{k}$ byte long
2. Encode $\widetilde{\mathbf{m}}'$ using the chosen $(\widetilde{n}, \widetilde{k})$ RSC outputting $\widetilde{\mathbf{m}}$, where $|\widetilde{\mathbf{m}}| = \widetilde{n}$
3. Remove the $\widetilde{k} - s$ zeros from $\widetilde{\mathbf{m}}$ (which is easily possible, as we chose a systematic scheme) and use the resulting $\mathbf{m}$ vector for channel input

Please note that – because we choose a systematic code – $\mathbf{m}$ will effectively be a concatenation of $\mathbf{m}'$ and vector of **parity check symbols**. Receiver can do the decoding the following way:

1. If receiver detects no erasures then assume $\mathbf{m}'$ is $\widehat{\mathbf{m}}$ (stripped off of the parity check symbols) and do the decoding step of the outer-code as usual
2. Otherwise reinsert $\widetilde{k} - s$ zeroes to $\widehat{\mathbf{m}}$ and do the RSC decoding step of the inner-code providing $\mathbf{m}'$ (right after the added zero symbols are removed)

If put into the perspective of the inner-code of our network, then the advantage of this approach is that both S and R can use a fixed RSC scheme, independently from the actual network parameters m, n and r chosen by the network designer.

To be able to use this scheme directly in our case, we have to make one more step. The reason for this is the fact that we have a varying set of pairs of network parameters m, r, while the scheme we choose has the fixed values of $\widetilde{n} = 255$, $\widetilde{k} = 223$ and $\widetilde{s} = 8$. Let us see a concrete example of how this goes in case the size of $\mathbf{m}'$ is 256, $m = 3$ and $r = 1$.

Because in case of the chosen RSC scheme $\widehat{r} = 32$ we have to proceed the following way with the **encoding**:

1. **Slice** $\mathbf{m}'$ into p disjunct pieces $\mathbf{m}''_{1..p}$, the size of which would be $(m - r) \cdot \widetilde{r} = 64$ bytes, each (where obviously $p = 4$)
2. **Pad** each $\mathbf{m}''$ with $\widetilde{k} - 64 = 159$ zero bytes, resulting in $\mathbf{m}'''_{1..p}$ vectors, each $\widetilde{k} = 223$ byte in size
3. **Apply** the chosen **RSC encoding** scheme onto each $\mathbf{m}'''$ vectors resulting in $\mathbf{m}''''_{1..p}$ vectors, each $\widetilde{n} = 255$ bytes in size
4. **Remove** $p \cdot (\widetilde{k} - (m - r) \cdot \widetilde{r}) = 4 \cdot 95 = 380$ **zeroes** from the $\mathbf{m}''''_{1..p}$ vectors (this is easily possible, because we chose a systematic RSC), yielding $\mathbf{m}'''''_{1..p}$ vectors, each $m \cdot \widetilde{r} = 3 \cdot 32 = 96$ bytes in size

5. **Slice all** $\mathbf{m}''''_{1..p}$ vector into m disjunct pieces of $\mathbf{m}''''_{1..p,1..m}$, each $\widehat{r} = 32$ bytes in size, where all last slice $\mathbf{m}''''_{1..p,m}$ are vectors of of **parity check bytes**
6. **Create channel input** as a concatenation of the $p \cdot m$ pieces of these slices as follows: $\mathbf{m} = \|_{a=1}^{p} \|_{b=1}^{m} \mathbf{m}''''_{a,b}$, being $p \cdot m \cdot \widetilde{r} = 384$ byte total

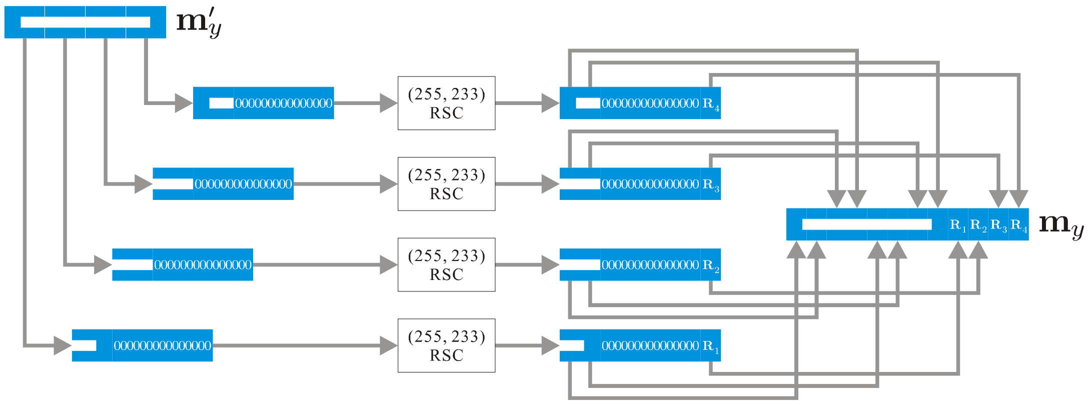


Fig. 14: (255, 223) RSC: second-stage encoding

Fig. 14 shows how this works in practice.

Due to the earlier explained properties of our scheme, if the number of inner-network branches affected by churn is r, the decoding of exactly r of the consecutive $p \cdot \widetilde{r} = 4 \cdot 32 = 128$ byte long parts of the $\widehat{\mathbf{m}}$ vector (more precisely r pieces of $\|_{a=1}^{p} \mathbf{m}''''_{a,k}$).

Fig. 15 demonstrates the complete inner-layer encoding in case RSC is used. This figure scales with Fig. 10, in the sense that these show the resulting vectors and their relative sizes in a network that has the same parameters.

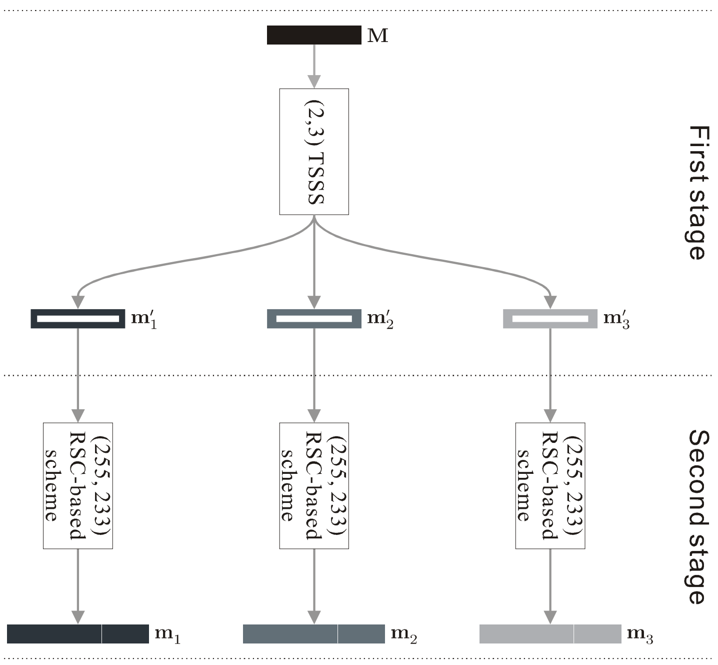

M
(2,3) TSSS
First stage
m'_1
m'_2
m'_3
(255, 233) RSC-based scheme
(255, 233) RSC-based scheme
(255, 233) RSC-based scheme
Second stage
m_1
m_2
m_3

Fig. 15: (255, 223) RSC: complete encoding process

Because in case of our scheme $r = 1$, one of these pieces may be affected by churn without loosing the capability of overall decoding of any of the $\mathbf{m}'$ vectors. Let us see how decoding works in case branch 2 manifests error. In case $\|_{a=1}^{p} \mathbf{m}''''''_{a,2}$ becomes undecodable via XOR, but R may still **decode** each $\mathbf{m}'$ the following way:

1. **Slice** $\widehat{\mathbf{m}}$ into $p \cdot m = 4 \cdot 3 = 12$ vectors of $\widehat{\mathbf{m}}''''''_{1..p,1..m}$
2. **Build vectors** $\widehat{\mathbf{m}}''''''_{1..p}$ from these pieces, while bearing in mind that the 2nd part of each has been erased (in the sense that those can not be decoded via XOR operations)
3. **Add** the removed **zero vectors** back to each $\widehat{\mathbf{m}}''''''$ providing $\widehat{\mathbf{m}}''''_{1..p}$
4. Use the decoding scheme of RSC, yielding $\mathbf{m}'''_{1..p}$
5. Remove the zeroes, providing the $\mathbf{m}''_{1..p}$ vectors
6. Recover $\mathbf{a}' = \|_{a=1}^{p} \mathbf{m}''_{x}$

Fig. 16 shows how this is done in practice.

Of course if no churn occurs, the decoding is much easier, as we used a systematic RSC, so simple rearrangement of $\widehat{\mathbf{m}}$ (and omission of the $p = 10$ pieces of parity

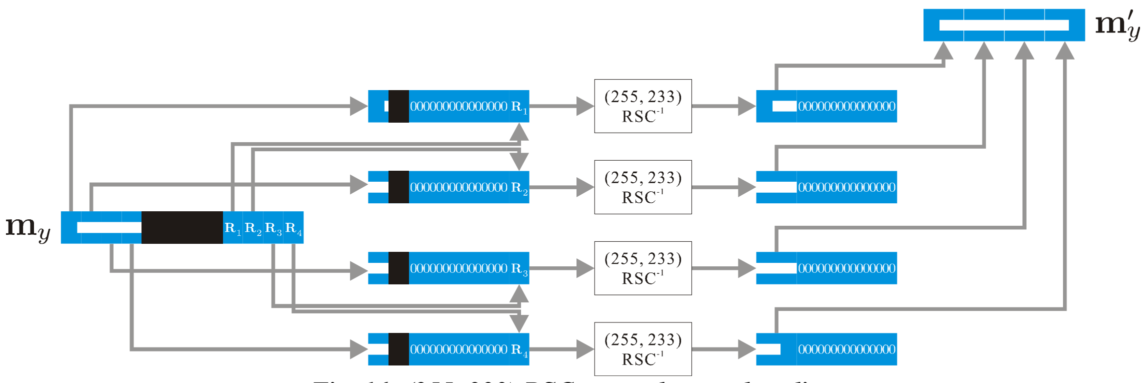


Fig. 16: (255, 223) RSC: second-stage decoding

check bytes) will provide $\mathbf{m}'$, so the decoding of the first-stage can be carried out directly. This can also be seen easily in Fig. 16.

Please note that in this example we managed to reach the rate of $R = 3/2$, because we assumed the size of each $\mathbf{m}'$ to be divisible by $\widehat{r} = 32$. If real life message sizes do not meet this criterion, padding each $\mathbf{m}'$ with random data can still provide rate expansions close to the ideal value.

Please note that the choice of RSC code parameters of $(255, 223)$ is just an example. We chose this simply because it has been well researched and referred to, therefore actual implementations of the necessary algorithms is widely available. See also the chapter Error: Reference source not found for further comments on this.

The last figure (Fig. 17) of this section shows both stages of the decoding scheme of the inner-layer with the parameters used in this chapter and under the assumption some/all nodes fail in the second branch.

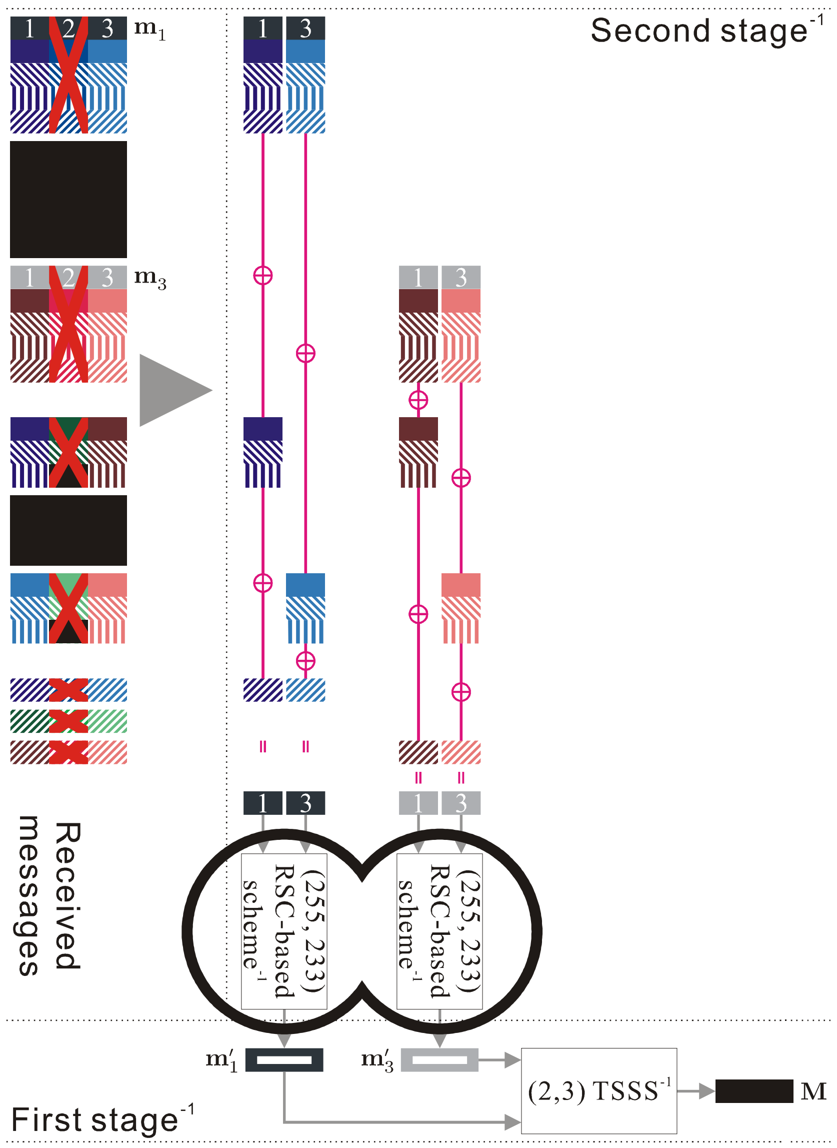


Fig. 17: (255, 223) RSC: complete decoding process

7 Analysis of Anonymity

In this chapter we will give a detailed explanation of:

- what **anonymity** relies on
- how attacker may attempt to **compromise** it
- what the attacker's **chances** to do so are in the ES and in the PS

The goal is to show to correlation between anonymity and churn resilience to give the users guidelines on how to select network parameters to have the desired level of both features at the same time.

7.1 Attack Model

Both the paper [2] and this MT will rely on the assumption that the actual communication that finally takes place between S and D (see Fig. 1) is encrypted. The details of that encryption is outside of the scope of both the ES and the PS, but it guaranties that an asymmetric key or one of the paired symmetric keys has previously been exchanged between them, so attacker may not have access to the hidden content of the final message.

However the system relies on no preliminary knowledge (keys) being shared between any other participants of the system and an attacker may attempt to compromise the anonymity of the sender S, by gaining access to the RI, this way connecting S to D. **Based on this, all our subsequent analysis aims at understanding the attacker's ways and chances to decode RI within the inner-layer.**

7.1.1 Attack Model in the ES

Due to the properties of the coding scheme they use (explained in chapter 5.2) – attacker may gain access to RI by becoming able to decode all $\mathbf{m}$ vectors of all branches. Let us look at how this may be achieved.

As each vector $\mathbf{m}_y^{(x)}$ collects RVs via XOR operations, a malicious $O_{y,x}$ needs cooperating helpers at very specific locations to be able to recode $\mathbf{m}_y$. By "very specific location" we mean the following rule: a malicious $O_{y,x}$ needs $m-1$ cooperating helpers at the following positions:

- If $x \leq n-2$: $O_{z,x}$ or $O_{z,x+1}$ or $O_{z,x+2}$
- If $x = n-1$: $O_{z,x}$ or $O_{z,x+1}$
- If $x = n$: only $O_{z,x}$

for $\forall z \in [1, m], z \neq y$

Theorem 2 of the paper [2] builds on a more complicated concept, involving a variable called r (not related to the r we are using within this MT), but we believe that their interpretation of the conditions can be further simplified as described above.

Only for reference the relevant part of the paper's theorem 2 – rephrased using our notation and terminology – is as follows: "$\ldots$ The piece $\mathbf{m}_y$ can be recovered if and only if there exists $s+1$ compromised nodes $O_{y,x}(x \in [t, t+s], s \geq 0, t \geq 1)$ that occupy $s+1$ successive position on branch y and for all branches z $(z = 1, 2, ..., m; z \neq y)$ there is at least one compromised node among nodes $O_{z,j}$ $(j \in [t, t+s+2])$ on each branch $z\ldots$"

The reasons for these conditions to hold are the following:

- To decode an $\mathbf{m}_y$, $O_{y,x}$ needs to obtain access to all the RVs that have been stacked upon $\mathbf{m}_y^{(x)}$
- $O_{y,x}$ holds y^{th} part of the RVs already in the form of $\mathbf{v}_y^{(x-1)}$ and $\mathbf{c}_{y,x-1,y}$
- Accordingly, he needs to gain access to the remaining RVs, that can be done in any of the following 3 different situations:
    1. $O_{z,x}$ is malicious: that node has $\mathbf{v}_z^{(x-1)}$ and $\mathbf{c}_{z,x-1,z}$
    2. $O_{z,x+1}$ is malicious: this node possesses $\mathbf{v}_z^{(x)}$
    3. $O_{z,x+2}$ is malicious: this node possesses $\mathbf{v}_z^{(x+1)}$. while additional information necessary for decoding was generated by $O_{y,x}$ itself (as $\mathbf{c}_{y,x}$)

Refer to Fig. 5 for a visual representation of this scenarios.

Any other node $O_{z,q}$ before or after this "window of 3 nodes per branch" either lacks information ($q < x$) or has a $\mathbf{v}_y^{(q)}$ that has gathered additional RVs via XOR operation ($q > x+2$). This property is inherited from the ES and is proven in that paper.

Fig. 18 shows 2 example scenarios when the malicious attacker $O_{2,1}$ may and when it may not decode $\mathbf{m}_2$ in case $m = n = 3$.

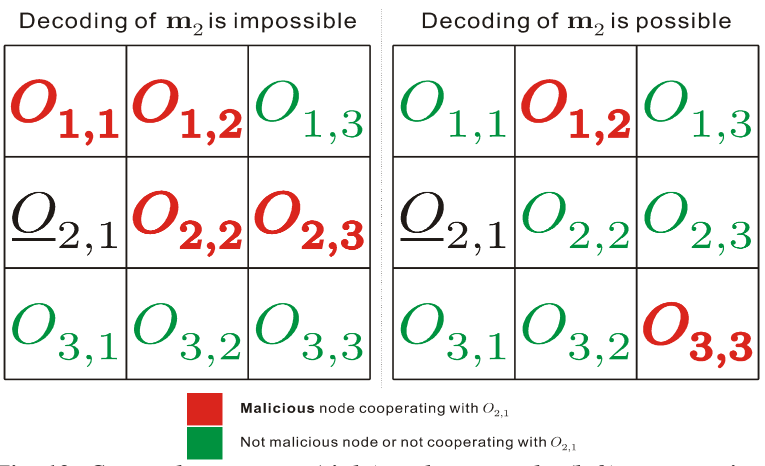

Decoding of $\mathbf{m}_2$ is impossible
Decoding of $\mathbf{m}_2$ is possible
$O_{1,1}$ $O_{1,2}$ $O_{1,3}$
$\underline{O}_{2,1}$ $O_{2,2}$ $O_{2,3}$
$O_{3,1}$ $O_{3,2}$ $O_{3,3}$
$O_{1,1}$ $O_{1,2}$ $O_{1,3}$
$\underline{O}_{2,1}$ $O_{2,2}$ $O_{2,3}$
$O_{3,1}$ $O_{3,2}$ $O_{3,3}$
🟥 **Malicious** node cooperating with $O_{2,1}$
🟩 Not malicious node or not cooperating with $O_{2,1}$

Fig. 18: Cases when* $\mathbf{m}_2$ *may (right) and may not be (left) compromised

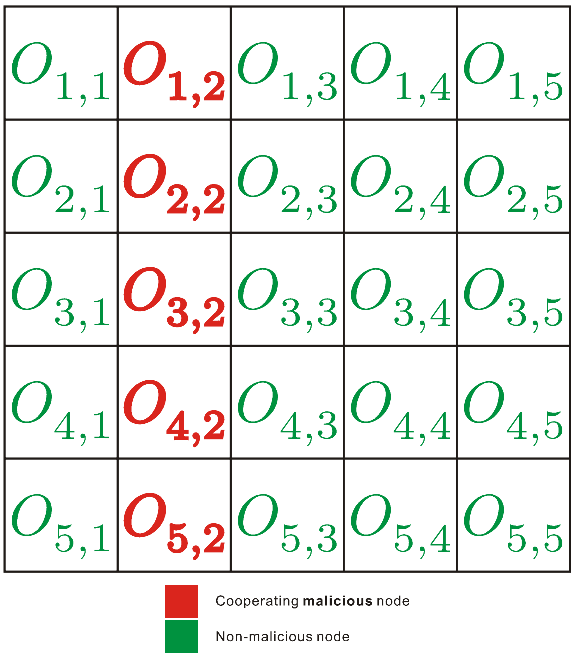
In the situation shown at the left side of Fig. 18, the number of malicious nodes is higher than on the right side, however access to $\mathbf{m}_2$ is not possible, due to the fact that $O_{2,1}$ does not have a **Cooperating Attacker** (CA) among $O_{3,1}$, $O_{3,2}$ and $O_{3,3}$. Even if the network was bigger and $O_{3,4}$ or subsequent nodes were malicious, $O_{2,1}$ would still be unable to do the decoding, for the same reason.

In contrast to this, the right side of Fig. 18 shows a case when a given attacker may become able to decode $\mathbf{m}_2$ vector. To decode RI, this is still not sufficient, so let us change our point of view and concentrate on decoding RI itself, as that would the ultimate target of any attacker.

It is obvious that in this situation it is not going to be a single node that does the decoding of $\mathbf{M}$ (equivalent to the RI), but a group of CAs. The actual decoding may take place anywhere (inside or outside of the network), but the following condition needs to be met either way: there must be **at least one** of the cooperating malicious **nodes in each branch** that satisfies the above-mentioned criterion.

Fig. 19 and Fig. 20 will present two cases where this condition is met and therefore the group of attackers gains access to RI. As explained above, this is a necessary and sufficient condition for an attacker to be able to compromise anonymity. Both figures, Fig. 19 and Fig. 20, show a network that has the parameters $m = n = 5$.

Fig. 19 represents the most obvious case, where attackers form a column. This is the situation when the number of attackers necessary to decode $\mathbf{M}$ is the minimum and this is being mentioned by the paper [2] as "$k = m$" (cf. the ES uses k to represent the constant number of attackers present in the mesh).



$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{1,4}$ $O_{1,5}$
$O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{2,4}$ $O_{2,5}$
$O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{3,4}$ $O_{3,5}$
$O_{4,1}$ $O_{4,2}$ $O_{4,3}$ $O_{4,4}$ $O_{4,5}$
$O_{5,1}$ $O_{5,2}$ $O_{5,3}$ $O_{5,4}$ $O_{5,5}$
🟥 Cooperating **malicious** node
🟩 Non-malicious node

Fig. 19: The easiest situation when $\mathbf{M}$ *may be decoded (ES)*

A more complicated situation is shown in Fig. 20, when attackers fail to occupy a full stage (column), therefore their numbers must also be bigger. As deducible, the minimum number of attackers necessary to decode will fall into the range of $[m, m^2]$, depending on the locations they manage to take.

As visible from the definitions, the chances (q) of a cooperating group of attackers (A) to fully decode RI depends in their numbers (k) and locations, according to the following rules:

- If $k < m$ then $q = 0$
- If $m \leq k \leq (m-1)n$ then $q \in [0,1]$
- If $m^2 \leq k \leq (m-1)n$ then $q \in (0,1]$
- If $(m-1)n + 1 \leq k$ then $q = 1$

7.1.2 Attack Model in the PS

As redundancy is added to the messages in the PS, attackers' chances do change compared to that of in the ES, The conditions explained in chapter 7.1.1 will turn into the following: a malicious $O_{y,x}$ needs $m - r - 1$ cooperating helpers (at the same po-

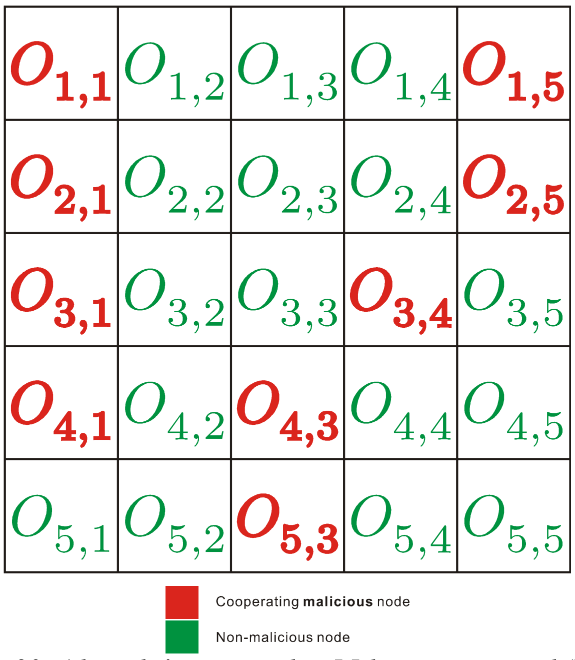

$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{1,4}$ $O_{1,5}$
$O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{2,4}$ $O_{2,5}$
$O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{3,4}$ $O_{3,5}$
$O_{4,1}$ $O_{4,2}$ $O_{4,3}$ $O_{4,4}$ $O_{4,5}$
$O_{5,1}$ $O_{5,2}$ $O_{5,3}$ $O_{5,4}$ $O_{5,5}$
🟥 Cooperating **malicious** node
🟩 Non-malicious node

Fig. 20: A less obvious case when $\mathbf{M}$ becomes exposed (ES)

sitions that were mentioned in chapter 7.1.1) in any of the $m - 1$ different branches $\rightarrow$ according to the construction rules, this will allow the decoding of sufficient part of $\mathbf{m}_y^{(x)}$ to be able to decode $\mathbf{m}_y$ itself.

Moreover, instead of m, $m - r$ CAs that satisfy the above mentioned conditions (gaining access to $m - r$ pieces of $\mathbf{m}$ vectors) are sufficient to decode $\mathbf{M}$ itself.

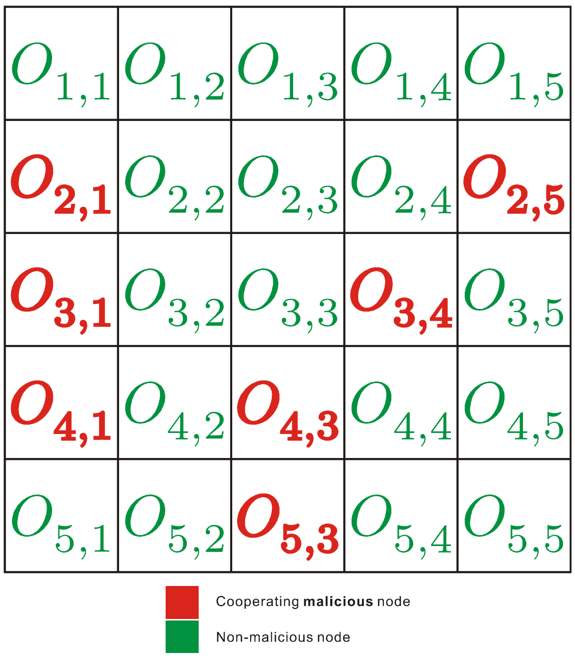

$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{1,4}$ $O_{1,5}$
$O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{2,4}$ $O_{2,5}$
$O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{3,4}$ $O_{3,5}$
$O_{4,1}$ $O_{4,2}$ $O_{4,3}$ $O_{4,4}$ $O_{4,5}$
$O_{5,1}$ $O_{5,2}$ $O_{5,3}$ $O_{5,4}$ $O_{5,5}$
🟥 Cooperating **malicious** node
🟩 Non-malicious node

Fig. 21: A case when* M *becomes decodable in the PS

Fig. 21 shows a case when RI may be decoded in the PS (as long as $r \geq 1$). It is interesting to note that if we compare two networks having the same parameters m and n, then:

- if something is decodable in the PS, that does not necessarily mean that it is also exposed in the ES
- but if something is breakable in the ES that necessarily means that it is also decodable in the PS

The details of how this affects affects anonymity in each case will be discussed in chapter 7.2 in details, but it is important to point out that what we really want to compare are not the two networks (one under the ES and the other using the PS) with the same parameters, but a given network that uses the ES to an other mesh that has the **same base parameters**, but higher number of branches, where the number of increment is expressed by well-known formula $m_{\text{PS}} = m_{\text{ES}} + r$.

7.1.3 Cooperating Attackers

Until now we kept using the term "**cooperating attacker**" (CA) but we feel this needs a bit of clarification and some discussions. CA means that the nodes:

- themselves participate in the forwarding mesh or have full and reliable access to the vectors received, generated and sent by those.

- have a full notion of the scheme – including its coding algorithms and parameters – and a strategy to share information between each other or store those at a central location for later processing.

It may easily happen that the presence of malicious nodes in the mesh is high, but those do not form a single network of CAs. In such situation each group belonging together (under the criteria of cooperation) has a chance to decode RI independently from the others.

An other – much less obvious – aspect of this is the **difficulty of decoding**. We will later show that the probability of successfully decoding RI correlates with the level of the presence of the CAs: probabilistically, the higher the presence, the higher the chance of decoding via XOR. Based on this, let us assume a cooperating group of attackers has managed to gain high presence in a pool of nodes. In this case:

- Malicious nodes will participate in large amount of the outer- and inner-layers formed
- The number of messages travelling for each $S \rightarrow D$ transfer is **cubic** or **quadratic**
- The messages received by the group is therefore very large
- It is easy to construct a scheme, where the actual index y of the vectors $\mathbf{m}_y^{(x)}$ is not directly readable, unless decoding of $\mathbf{m}_y$ is done. This may slightly complicate the decoding algorithm at each R, but – due to the small number of messages each of those receives – the complexity is bounded
- Each sender node of the outer- and the inner-layer knows only the identity/ies – network address(es) – of its successor(s) (and vice versa), but not the nodes that are in the successive stages
- If decoding of $\mathbf{m}_y$ is attempted within the mesh, then – as explained in the chapter 7.1 – the algorithm depends on relative location of the nodes
- The number of possible combinations further increases in case of the PS (see chapter 7.1.2)

- Nodes may – and in case of a regular network will be – part of multiple networks and sessions

The consequence of this is that attackers would have a combinatorially large amount of combinations of messages to permute through, each time attempting to decode RI, to be able to identify the to-be hops R of the outer-layer. As the size of the network grow, this will put attackers into the following situation:

- Low presence keeps the chances of the attackers very low (see chapter 7.2.2)
- In case of high presence attacker soon face computationally infeasible situation

As our interest lies in finding the upper bound on some CAs' chances to decode RI, subsequent analysis of the extent of the difficulties attacker face when attempting to compromise anonymity is outside of the scope of the MT.

7.2 Simulations

There are multitudes of strategies a node S may follow when trying to build its inner-layer. Unless we make any special assumption on S having helpers, the best strategy for S is to **randomly choose** $m \cdot n$ nodes from the pool of available nodes. By *best* we mean the fact that this is the situation when in a simple network attackers **have no means of biasing** S towards acquiring favourable (from the anonymity's point of view: unfavourable) presence and/or location within the network. By *simple network* we mean the fact that no additional mechanism (e.g. Reputation System) is present that may allow S to know or to make assumption on who the malicious nodes are.

For the reason the paper [2] has not mentioned any such technique to be implemented, we assume that S will be aware of sufficiently large amount of peer nodes that are ready to serve the purpose of inner- or outer-layer relay nodes. In a real-world implementation *being ready* means that the node is online, it is running the client that executes the anonymity protocol and it will do its best to fulfil its purpose (which ultimately depends on whether it is a normal or malicious nodes).

During normal operation of the network, S will select $\ell \cdot (m \cdot n + 1)$ different relays from among these nodes and attempt to contact D through them via the known techniques. Because of this, our simulation will follow this behaviour of the network. Before presenting the results, let us spare some words on the details of the two different types of simulations we have run and of the our motivation behind each.

Due to the fact that the ES is fairly vague with regard to the level of explanation in provides on the details of its simulations, our first target was to reproduce the results of the paper to be able to compare ours to them. The details of this attempt is discussed in chapter 7.2.2.

Based on this we have established high level of confidence that both the original paper and we interpret anonymity the same way and that our sets of simulations also yield the same results under the same initial conditions. Based on this, we also run a series of simulations different in nature from that of the paper. The reason for this is that – we believed – our model reflects the real-life situation better and gives a wider perspective of the correlation between all existing network parameters of the inner-layer, such as m, n, r and p (proportional presence of cooperating malicious nodes in the network). The details of the methods and results of these simulations shall be described in chapter 7.2.3.

7.2.1 Base Setup

In all cases our simulations have consisted of the following steps:

1. We **fixed parameters** m and n, set $r = 0$ (referring to the ES) and iteration value $i = 10.000$

2. Under these conditions our simulation engine **generated** i independent **inner-layers** with specific type of presence of CAs at **random locations**. The type of presence was either **fixed** at k (see chapter 7.2.2) or p **proportional** (see chapter 7.2.3)
3. We evaluated whether CAs would be able to decode $\mathbf{M}$ in case of the generated network layout.
   - For the ES ($r = 0$) this evaluation has been done based on the criteria described under chapter 7.1.1
   - In case of the PS ($r \geq 1$) we used the criteria discussed in chapter 7.1.2
4. The basic parameters m, n and r have been changed according to the type of simulation and execution has been restarted from step 2
5. The value of i has been **increased** by about one magnitude and the steps has been rerun starting from step 2

During the simulations i has taken the values of $i = 10.000$, $i = 100.000$, $i = 300.000$ and finally $i = 1.000.000$ so that we would be alarmed by lack of convergence of the simulations' output. As the results have converged sufficiently already at $i = 10.000$, we expect our simulations to reflect the real-life scenario close enough to be able to serve the basis for evaluation. All the results presented shortly are based on the case of $i = 1.000.000$ only. Finally the simulation engine has saved all its values into binary and text files for later processing.

In the subsequent chapters we will refer to **confidentiality** $c \in [0, 1]$ being the chance that attackers can not decode RI. In other words $c = 1 - q$. The original paper also refers to confidentiality under the abbreviation $\mathrm{S}_{\text{AC-ITNC}}$.

7.2.2 Comparison to the Paper's Results

Before presenting the simulation results we would like to shortly discuss the following of the assumptions the paper [2] made: according to their model the first stage of nodes ($O_{1..m,1}$) in each inner-layer is not malicious.

We believe that either **such assumption is groundless**, or **makes the areas** where the ES can be used very **limited**, simply because in a general P2PN most of the peers do not have the means to guaranty such assumption to be valid, while failing to meet it would greatly declare the chances of S to maintain anonymity.

Due to this – until proven otherwise – **we made the evaluation of the PS without this assumption** being present. However only at this stage, when trying to compare the results of the paper [2] and those that were calculated by our simulations, we will maintain this assumption.

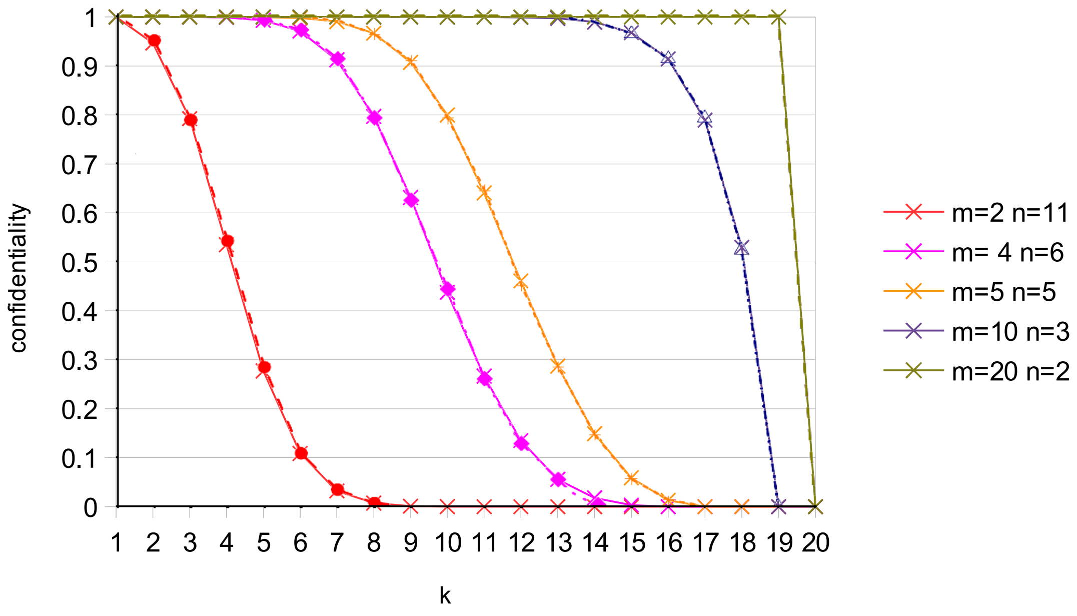


Fig. 22: Validating proof of our simulation engine's correctness

To be able to compare the performance of the anonymity of the two schemes, we first attempted to reproduce the results presented in the paper under their "Fig. 3". For this purpose we have executed the earlier explained simulation with matching values of $m, n, r = 0$ and $\forall k \in [1, 20]$, while generating only situations when attacker is unable to acquire position at stage 1. Here k stands for the fixed number of malicious nodes.

Fig. 22 does this comparison of our results with that of the paper (shown by their "Fig. 3") through laying the two sets of curves on top of each other. Out curves are shown in the legend and are displayed with thin lines bearing X marks. The paper's curves are the dashed thick lines not listed in the legend. To avoid misunderstanding, we used similar color for every pair of curves that were generated under the same set of network parameters.

As visible in Fig. 22 there is a perfect overlap between the two sets of curves, sufficiently showing that our simulation environment matches with theirs.

7.2.3 Our Set of Simulations

Based to the findings of and conclusions drawn in chapter 7.2.2, we have decided to detach from the original simulation technique and do our own independent set of analyses. **The single idea in mind was to visualize the correlation between level of CR and the strength of the anonymity** provided by the system in a real-world environment.

Using the output of such simulations, a network designer would be able to build a notion of the underlying mechanisms and make educated design decisions regarding the network's basic parameters m, n and r so that both his expectation regarding error resilience and confidentiality would be met.

We designed simulations that would compare the level of confidentiality provided by the ES and the PS with different redundancy values r under the same network configuration m and n and given that the level of malicious nodes' presence in the pool of available relay nodes is p. In addition – and as hinted earlier – we abandoned the paper's idea of making assumptions on the whereabouts of the malicious nodes, so in our simulations CAs may take any position in the inner layer's relay mesh with equal probability (p). Under these conditions we have executed a full set of simulations – according to the explanation of chapter 7.2.1 – with p values ranging for $p = 0.03$ up to $p = 0.45$ using the stepping $\Delta p = 0.03$ (resulting in 15 steps each).

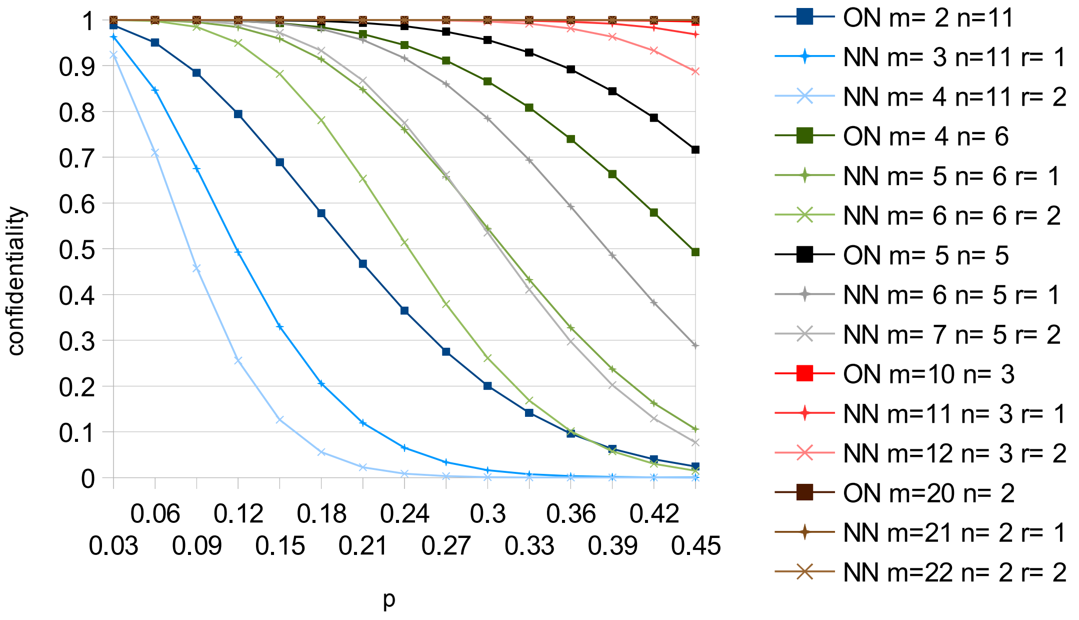


Fig. 23: Comparison of confidentiality in the ES and in the PS

Fig. 23 shows the visualisation of this comparison in 2D for a selected set of m, n pairs. These pairs have been selected so that those would represent deep ($m \gg n$), rectangular ($m \approx n$) and also shallow ($m \ll n$) inner mesh networks to get a first impression on how these variables affect confidentiality. Darker colours (with **boxes**) show the anonymity performance of the ES ($r = 0$), while the lighter counterparts of those represent the same for the PS ($r = 1$ with **diamond** and $r = 2$ with **cross**).

To be able to draw a wider -spectrum conclusion on the correlation between the anonymity and all the parameters of the network, we have also created 3D images. The following 3 figures demonstrate the level of confidentiality in case of all values of $m \in [2, 20]$, $n \in [2, 11]$ and $r = 0$ (Fig. 24), $m \in [3, 21]$, $n \in [2, 11]$ and $r = 1$

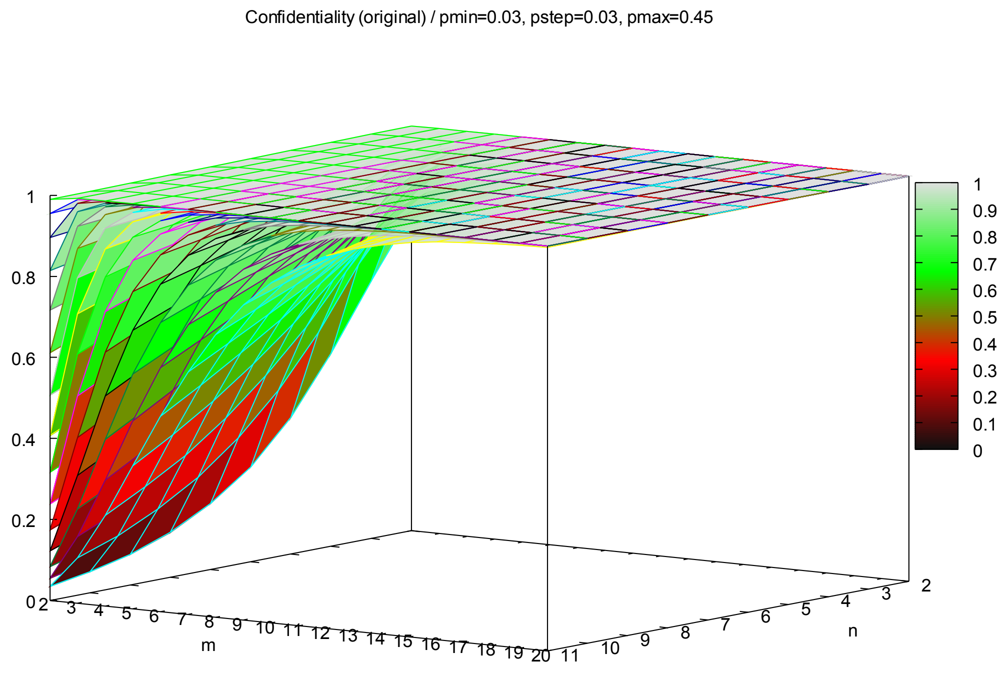


Fig. 24: Confidentiality in the ES

(Fig. 25) and $m \in [4, 22]$, $n \in [2, 11]$ and $r = 2$ (Fig. 26) while the average level of attackers (p) goes through the transition mentioned above.

Due to difficulty of representing 15 surfaces in a single figure, the 3 graphs need slight explanation. Each surface shows the level of confidentiality for specific value of p and for all the possible combinations of m and n. What can also be seen is that – as expected – the l**evel of confidentiality strictly monotonically decreases as the level of attackers' presence increases**.

To allow the viewer to make a comparison of the 3 situations at a fixed value of of p, we will also present 3 figures showing the behaviour of the confidentiality when $p = 0.21$. shows the case for the ES (Fig. 27), while and demonstrate PS with $r = 1$ (Fig. 28) and $r = 2$ (Fig. 29), respectively.

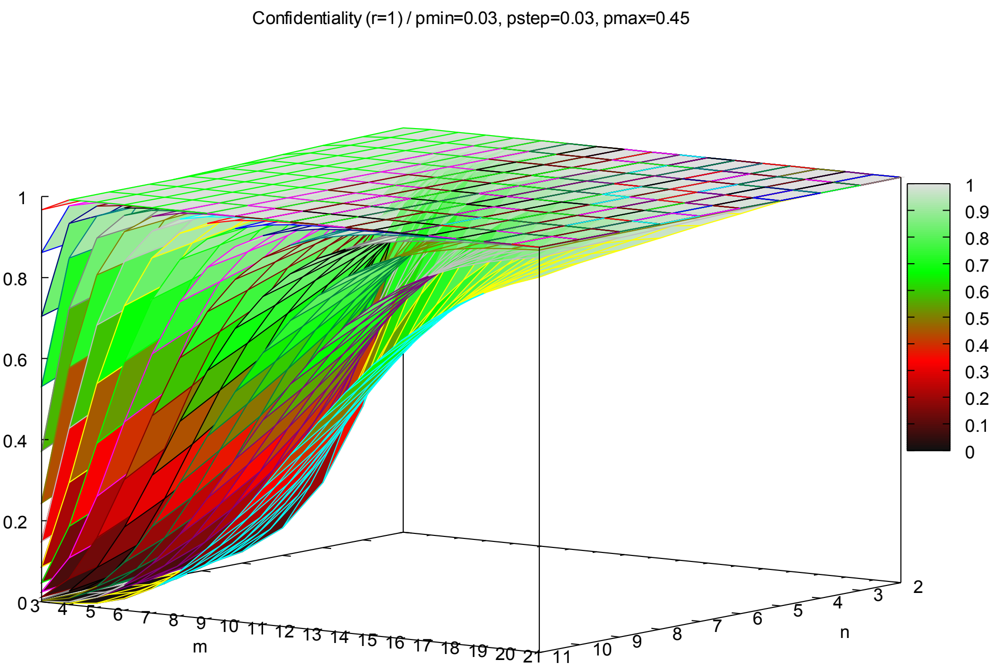


Fig. 25: Confidentiality in the PS (r=1)

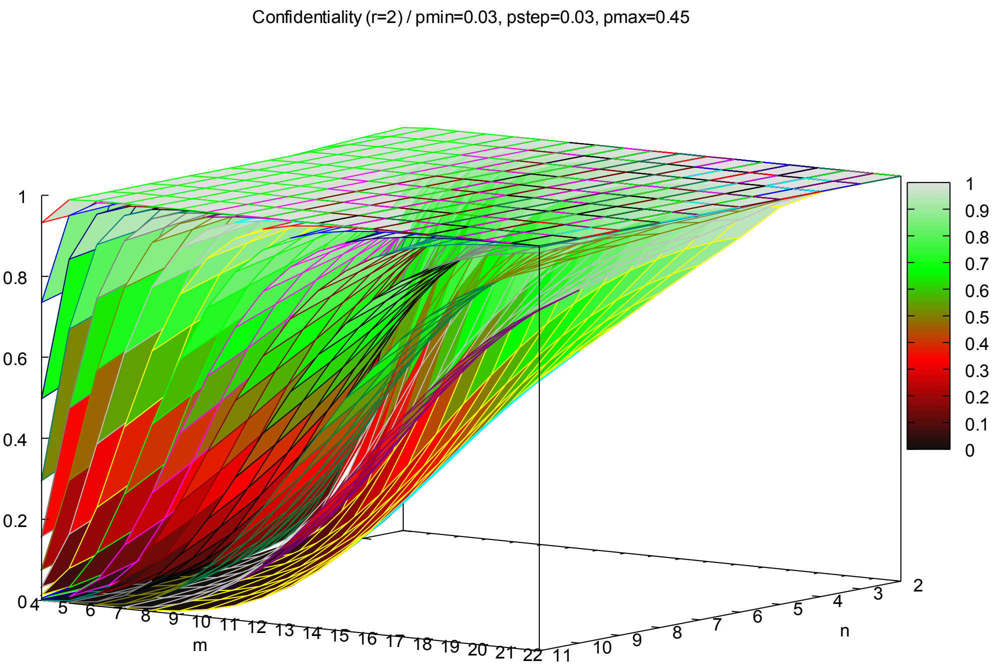


Fig. 26: Confidentiality in the PS (r=2)

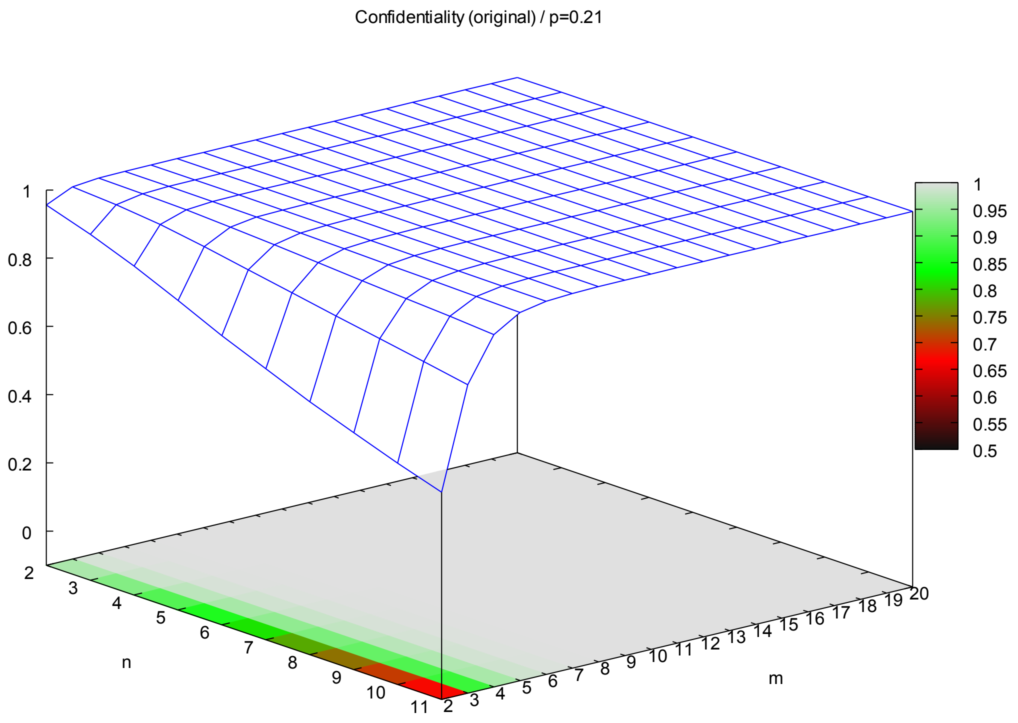


Fig. 27: Confidentiality in the ES (p=0.21)

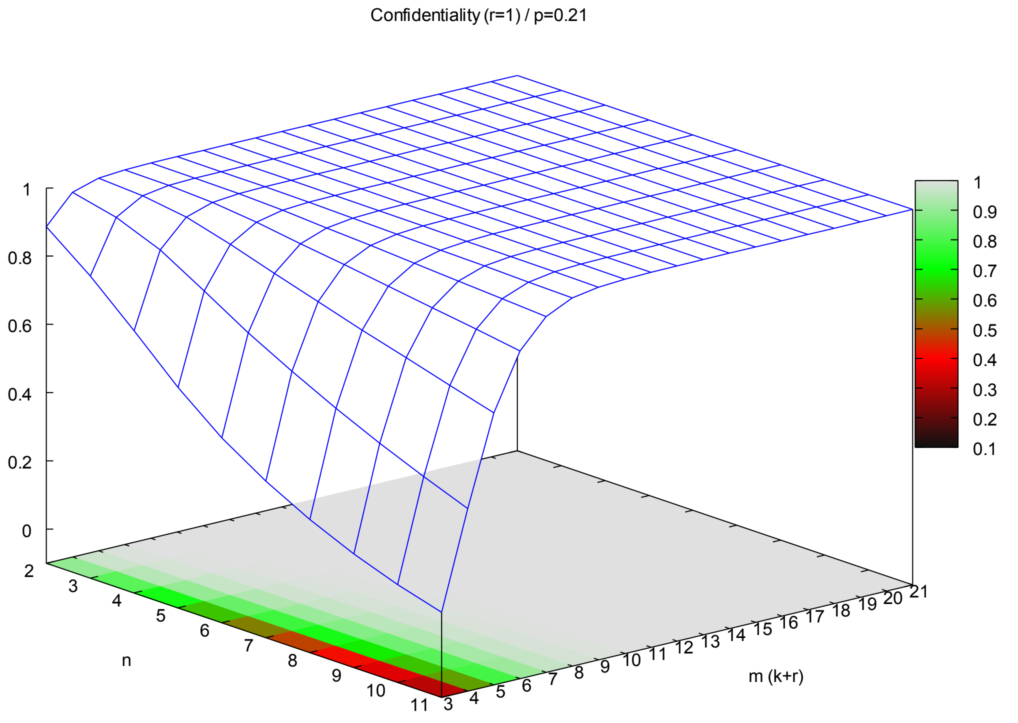


Fig. 28: Confidentiality in the PS (r=1, p=0.21)

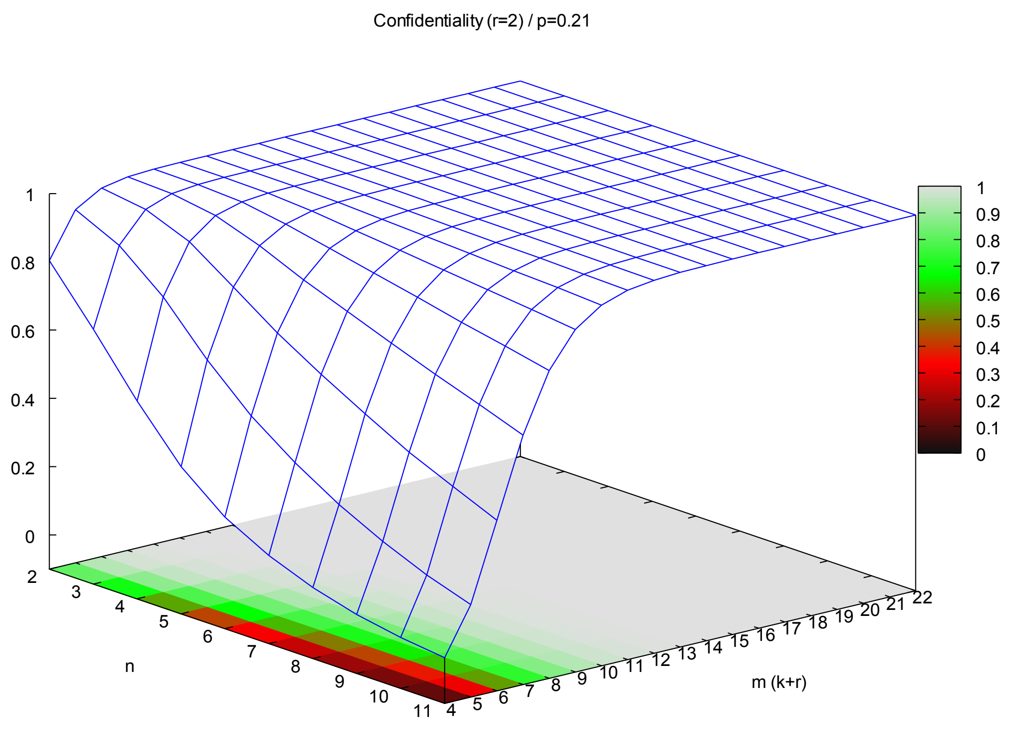


Fig. 29: Confidentiality in the PS (r=2, p=0.21)

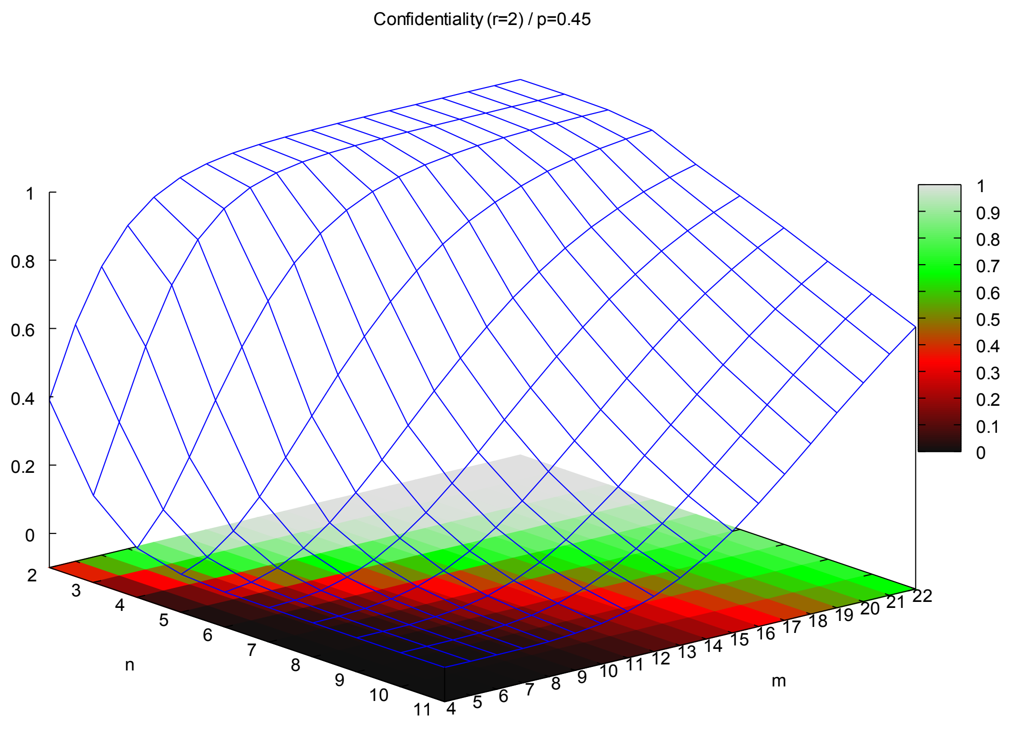


Fig. 30: Confidentiality in the PS (r=2, p=0.45)

Due to the space constraints, we will not like to burden this MT with figures any further, except for Fig. 30, which shows the case of $r = 2, p = 0.45$. The reason for including it is the presence of an interesting phenomenon (discussed in chapter 7.2.4).

7.2.4 Summary and Interpretation of the Simulation Results

Based on the simulation results explained and shown in the chapter, we would like to draw the following conclusions regarding the correlation between CR and level of offered confidentiality:

1. **Churn resilience is inverse proportional to confidentiality**
2. However **even** in case of strong churn resilience and very **high level of attacker presence** (Fig. 30), **there exist sufficient number of network configurations** ($n \ll m$) **when the level of confidentiality is very high**

As both CR and confidentiality are desirable properties, point 1 presents a **trade-off** relationship between the two. In other words network designer has to decide network parameters along this trade-off to achieve the required level of both. Fortunately point 2 says that high level of confidentiality can be achieved even in case of churn resilience and high level of attacker presence.

7.3 Further Comments

In this chapter we would like to draw attention to two relevant observations we made during our research.

7.3.1 Partial Decoding

Both the paper [2] and this MT analyse only the two situations when a given slice m is **fully** decodable or when it is **not** decodable **at all**. All modelling of the attacker's chances has been done based on this to be true. However we believe there exists a 3rd alternative the provides some chances to the attacker to compromise anonymity. An example of this is shown by Fig. 31 (cf. Fig. 20).

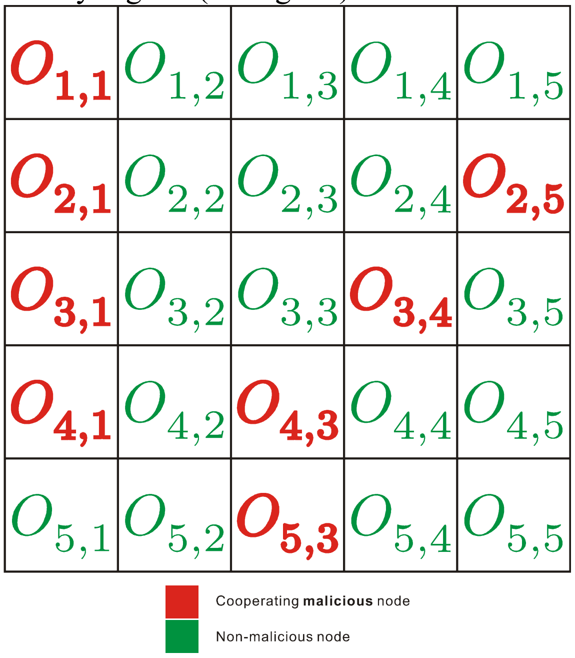

$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{1,4}$ $O_{1,5}$
$O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{2,4}$ $O_{2,5}$
$O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{3,4}$ $O_{3,5}$
$O_{4,1}$ $O_{4,2}$ $O_{4,3}$ $O_{4,4}$ $O_{4,5}$
$O_{5,1}$ $O_{5,2}$ $O_{5,3}$ $O_{5,4}$ $O_{5,5}$
🟥 Cooperating **malicious** node
🟩 Non-malicious node

Fig. 31: A case of partial decoding

The only change we applied to this compared to Fig. 20 is the removal of $O_{1,5}$ from the group of CAs. Because of this, $O_{5,3}$ will be unable to decode the 1st $1/5\text{th}$ of $\mathbf{m}_5$ but $4/5\text{th}$ of it and all the other 4 $\mathbf{m}$ vectors will be available for the attacker.

The ES assumes failure of compromising anonymity (in other words they assume presence of full confidentiality) in this situation, while we believe that the attacker **did acquire some information on RI** and has a higher chance of decoding RI compared to the situation when he got no message parts at all. For the attacker to decode MI, there would be nothing more to do than attempt decoding with all the possible $2^{\frac{|\mathbf{m}|}{m}}$ values of the missing part, which is much less than the usual case of $2^{|\mathbf{m}|}$ (that represents the number of all the possible values of MI).

In the light of the discussion outlines in chapter 7.1.3, this is possible (feasible) only in very special cases, when the network parameters permit it or the attacker – for any reason – decides to spare no costs for a duration of time. While in most of the cases decoding via this cumbersome way is not feasible for an attacker, we still believe that we should discuss some solutions to this problem.

The **theoretical approach** is to apply coding techniques to prevent attacker gaining information on the content of the $\mathbf{m}$ vectors. As attacker may gain access to additional information over $\mathbf{m}$ in steps (with the very low granularity of $\frac{|\mathbf{m}|}{m}$), we are looking for a coding technique that would prevent access to $\mathbf{m}$ as long as less than $\frac{m-r}{m}\mathrm{th}$ of it is decodable ($r = 0$ in case of the ES). The most obvious such scheme is a TSSS itself. For example if we decide to apply the **second stage** coding step demonstrated in Fig. 10 using (5,5) TSSS scheme, then the fact that $O_{1,5}$ is not malicious will leave the group of attackers without any information on the RI. Needless to mention that in such case the excessive expansion of the total size of messages travelling over the network discussed in chapter 6.4.1 will take effect. It is always up to the designer of the actual network to fine tune the system parameters along this other trade-offs between the level of anonymity and network economy.

If perfect secrecy is not required, a much more **practical solution** would be to:

1. **keep RI** as **small** as possible (by design)

and

2. to refrain from embedding any validation code (e.g. CRC) into either $\mathbf{m}$ or $\mathbf{M}$

Point 1 will limit the inter-dependency present within the RI, this way also limiting the attacker's chances of gaining side-information. If this point can not be directly met (e.g. IPv6 addresses do show a predictable pattern), then an appropriate compression algorithm may be applied to RI – before encoding it into $\mathbf{M}$ – to hide its statistical properties.

Point 2 serves a similar purpose: trying to make the attacker unable to rule out too many of the outputs of the iterative decoding it did, based on some validation code that would indicate erroneous output. Unfortunately this conflicts with the mechanisms of TR. In case TR is required, a validation code may still be kept, but done through a computationally expensive algorithm.

7.3.2 The First/last Nodes of the Outer- and the Inner-layers

Until now both the paper [2] and we drew equality between decoding all the RI and anonymity being conquered. In this chapter we would like to point out that this is not necessarily the case.

Let us look at Fig. 1 again. What will an attacker who managed to decode the RI sent to R_1, R_2 and to R_3 know? Based on RI_1, attackers will know that somebody wants to use R_2 to send something. Similarly RI_2 will tell that R_3 is involved, while RI_3 identifies D to be a node in the chain. If the parameter ℓ is fixed ($\ell = 3$), then the attackers already know not to look any further for RI.

Based on the information acquired until now, attackers know that from among R_2, R_3 and D one is the destination of a one-way anonymous messaging attempt, but the attacker does not – necessarily – know their order or the identity of S.

To gain order information, attackers have to be members of the last stage of each mesh of the inner-layer. In other words at least one of the $O_{1..m,n}$ nodes of each ℓ mesh has to be malicious. This holds, because – due to the network's structure – members of the last stage are aware of their location, due to the fact that those send all their messages to one single destination (R).

By gaining access to all RI and by taking positions in the last stages of the inner-layers, attacker is able to identify the routing chain $R_1 \to R_2 \to R_3 \to D$ and the final destination D in it, but this still does not identify S. Knowing neither S, nor what it wants to sent to D leaves the sender anonymity intact.

There are the following **two ways** an attacker may still compromise anonymity, given that it already met the conditions explained above:

1. R_1 **is malicious**: in this case the attacker completes the chain of OR routers and knows that S is contacting D
2. **Any of** the $O_{1..m,1}$ nodes in any of the inner-layer networks is malicious: due to the network design, the first stage of the inner-layer mesh knows about being contacted directly by S (simply because they receive their inputs from one single node: S itself)

Due to these additional limitations being in place, the earlier results presented in this chapter (especially chapter 7.2.3) showing the strength of anonymity should be considered to be **lower bounds on confidentiality**.

8 Tampering Resilience

In this chapter we will define the terms **Tampering** and **Tampering Resilience** (TR) as opposed to churn, and propose a very simple scheme that allows us to extend the established CR to deal with tampering as well.

8.1 Tampering

Within this MT we define tampering the following way: Tampering is the **purposeful action** of malevolent node(s) that aims at **disrupting the communication** of the RI distribution network. Based on this definition, The main difference between churn and tampering is the fact that churn occurs as a result of error situation that can not be attributed to purposefulness.

The tools of a tampering node $O_{y,x}$ may fall into several categories, such as:

1. Accepting incoming messages, but **not sending anything** in return within the available time frame
2. Sending **multiple** outputs
3. Forwarding **corrupted** $\mathbf{m}^{(x)}$, $\mathbf{v}^{(x)}$ and/or $\mathbf{c}_{y,x,1..m}$ **vectors**
4. **Any** possible **combination** of the above

Please note the similarity and differences between churn and tampering and refer to chapter 6.2.5 as well.

8.2 An Extension of the PS to Cover Tampering

There are at least two basically approaches to implement resilience against attacker's attempt to disrupt the communication. One is computationally more demanding but requires no changes to the system while the other would manifest a more complex design but it may be less strenuous on R.

Within this MT we will present the first approach by proposing a solution that implements TR based on the already existing CR techniques. The basic idea is the following:

1. At S use an error detection coding scheme, such as **Cyclic Redundancy Check** (CRC) code or **Hash Function** (HF) [7] (whichever is more convenient on the given platform) to create a fingerprint of the RI
2. Before slicing RI, embed its **fingerprint** into $\mathbf{M}$
3. At each R do the decoding as explained earlier
4. Verify the fingerprint to be correct
5. In case of mismatch, simulate churn $1..r$ branch failures all possible ways until the fingerprint of the decoded $\widehat{\mathbf{M}}$ produces a valid fingerprint

We call this technique computationally costly due to the fact that step 5 requires at most $\sum_{x=1}^{r} \binom{m}{x}$ decoding cycles. Needless to say that by choosing an otherwise light-weight inner- and outer-code [1] [12] this stress on R can be eased considerably.

This scheme can also be integrated into our CR technique, the following way:

- If there has been any (e.g. f) node (in other words branch) failures during the decoding step (step 3), then decrease r by f
- Exclude those branches that have been detected to have failed from the permutation used at step 5

9 Summary and Final Conclusions

In this MT we have **presented an existing one-way anonymous communication system** [2] that provides advantages over earlier solutions [14].

Next we have **pointed out its** biggest **weakness being the lack of resilience against node failure** (churn). We have shown that a **double-redundant coding scheme** exists that can provide predictable **churn resilience**.

Based on the new scheme, we made **detailed analysis of** its performance with respect to the level of **anonymity** it can provide **in case of** the presence of **cooperating malicious nodes**.

Finally we have introduced the term **tampering** being the purposeful and deliberate action of attackers to **disrupt** anonymous **communication** by whatever means they possesses as members of the forwarding mesh of the inner-layer. We have to pointed out the similarities and differences between tampering and churn and proposed a simple decoding scheme that would provide protection against tampering.

Based on the outcome of our research we believe that:

- it is to the advantage of the existing design to be extended with our error resilience scheme
- our work gives the theoretical and quantitative support a network designer needs to be able to make educated design decisions (selecting the network parameters ℓ, m, n and r) along the **trade-off** present between the level of resilience and anonymity provided by the system

10 Future Work

The paper describing the ES [2] does not give details on the method through which the inner-layer nodes are supplied with the necessary information – namely the network address of the node(s) in the next stage – to be able to execute the necessary forwarding algorithms. We believe that there are at least the following two improvements that must be made in this area:

1. The obvious: describe the missing design principles
2. Analysis:
    - Either show that such additional steps would not compromise the established properties of the scheme
    - Or point out the affected properties and analyze the effects of those on the overall system's anonymity properties

In addition to the need for clarifying important details left undisclosed, we believe there are at least three areas where the system may be further improved. The details of these are discussed in the following three chapters. The detailed analysis of each may be the subject of future research.

10.1 Replacing Onion Routing

As mentioned in chapter 5.1 the ES proposes the use of "traditional onion routing" as the coding scheme for the outer-layer. While OR is one of the first routing technique that supports hiding of identity, we believe that replacing it by more specific and lightweight solutions would bring advantages to the overall system performance. In a typical anonymous network OR is used to support at least the following two features:

- **Hiding** the **correlation** between a relay node's incoming and outgoing messages: this is necessary, when attackers have the capability of snooping on all the lines of all the nodes in the mesh. Such capability are very rarely assumed to be present, simply because such AP2P solutions are meant to run on very large network (such as the Internet itself), where no entity holds both the access and computation power to be able to receive and process all the messages. If this assumption were to be present in the ES, then the inner-layer would fail anyway, simply because – as discussed earlier in details – a group of attackers being able to occupy the first stage is capable of decoding RI. However if attackers are anyway capable of occupying all the positions of the outer-layer, then anonymity is broken, independently from other circumstances.
- **Hiding** the **routing information** (network address of the subsequent hops) from other members of the OR chain: This is a feature that is completely unnecessary in the ES, because routing information does not travel via the same path, but is distributed by the inner-layer

Removing OR would decrease the computational power required from the nodes of the outer-layer, allowing low-power entities to serve the purpose of relay in this network. Increasing the number of non-malicious nodes is always welcome, as – unless it is easy to take control of such nodes – such action means that a given group of CAs will loose relative presence..

If the size of the actual secret that travels between S and R has recognizable size distribution, then it may be necessary for each relay to pad the outgoing messages with random data.

10.2 Tailoring the Routing Information

We believe that with a specific design consideration in mind, it is possible to further increase the level of anonymity this scheme may offer. The ES assumes that anonymity is broken if the RI is decoded. While we believe this is not necessarily the case (see details in chapter 7.3.2), we do see a chance to definitely decouple the two, by designing the routing information a specific way. We see a chance that in case the RI follows the rules listed below, even decoding of all ℓ pieces of RI gives a chance to S to hide its identity:

- Make the RI a **pair of flow-id** and **network address** (e.g. IPv4 or IPv6)
- Use random flow-id for each relay R separately
- Upon receiving such RI, expect the relays to maintain this pair of flow-id and network address in their memory until relevant OR routing request arrives or a timer expires

Now assume that a group of attackers manages to take positions in the inner-layers in a way that they succeed in decoding all ℓ pieces of RI, but happen not to be able to occupy positions of the first first stages. In such situation attacker will not have information on either the sender S, nor the final destination D, as the mismatching flow-ids would not give grounds for being able to arrange these nodes into a chain of OR relays belonging to the same flow. While in a small network (where the number of concurrent transactions is small) the attackers will have a good chance to assume who is D from among $\ell + 1$ network addresses they uncovered, such chances would diminish in a bigger and busier network. By choosing the network parameter m low, network designer has a means of limiting the attackers' chances of taking hold of some of the first stages. We believe that in such network the attacker's only chance of compromising anonymity is through occupying all positions $R_{1..\ell}$, but this needs further research.

10.3 Probabilistic System Parameters

Following the thought pattern of chapter 10.2 we see a chance of being able to further decrease the attackers' chances of reconstructing the OR path by making the length of it (ℓ) be determined session-by-session by probabilistic techniques. An example of this could be the following scheme:

- Let ℓ be determined or influenced by a Bernoulli experiment with $p = 0.5$
- To allow the user to set a limit on the generated number

Implementation of such technique is easy, based on the random vector generator discussed in chapter 5.2 that is anyway to be present in the system (for the sake of the RVs)

Using this technique user may always have the sanctuary of **plausible deniability** by being able to say “that is not the node I contacted”, but this item also needs furhter attention.

11 Acknowledgements

First of all I would like to express my appreciation towards my advisor Prof. Hiroyoshi Morita, who never ceased to remind me that there was always more to be explored. I hope this enthusiasm for meaningful research will linger with me in the future.

I am equally lucky to have met Assist. Prof. Todorka Alexandrova who has not only followed my work closely, but also took part in it. Her support stretched beyond the realm of the academia and helped me to better understand Japan.

Last but not least thanks to Péter Zs. Schmidt who was there whenever a mathematical puzzle took shape.

12 References

[1] J. Kurihara, S. Kiyomoto, K. Fukushima, Toshiaki Tanaka,
"A Fast (3, n)-Threshold Secret Sharing Scheme Using Exclusive-OR Operations",
IEICE Transactions on Fundamentals of Electronics, vol. E91-A, issue 1, pp. 127-138, 2008.

[2] W. Wang, G. Duan, J. Wang, J. Chen,
"An Anonymous Communication Mechanism without Key Infrastructure based on Multi-paths Network Coding", *proceedings of IEEE GLOBECOM 2009, 2009.*

[3] Anonymizer, http://www.anonymizer.com

[4] P. F. Syverson, D. M. Goldschlag, M. G. Reed,
"Anonymous connections and onion routing",
IEEE Symposium on Security and Privacy, pp. 44-54, 1997.

[5] T. Alexandrova, G. Huszák, H. Morita,
"Churn Resilience in Network Coding-based Anonymous P2P Systems",
ISITA 2012.

[6] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for Web transactions",
ACM Transactions on Information and System Security, vol. 1, no. 1, pp. 66-92, 1998.

[7] R. E. Blahut "Digital Transmission of Information",
Addison-Wesley, ISBN 0-201-06880-X

[8] A. Shamir, "How to share a secret", *Communications of the ACM 22, pp.612-613, 1979.*

[9] Napster, http://www.napster.com

[10] T. Ho, D. S. Lun, "Network Coding, an Introduction",
Cambridge UP, ISBN 978-0-521-87310-9

[11] M. Médard, A. Sprintson, "Network Coding, Fundamentals and Application",
Elsevier, ISBN 978-0-12-380918-6

[12] J. Kurihara, S. Kiyomoto, K. Fukushima, Toshiaki Tanaka,
"On a Fast (k,n)-Threshold Secret Sharing Scheme",
IEICE Transactions on Fundamentals of Electronics, vol. E91-A, issue 9, pp. 2365-2378, 2008.

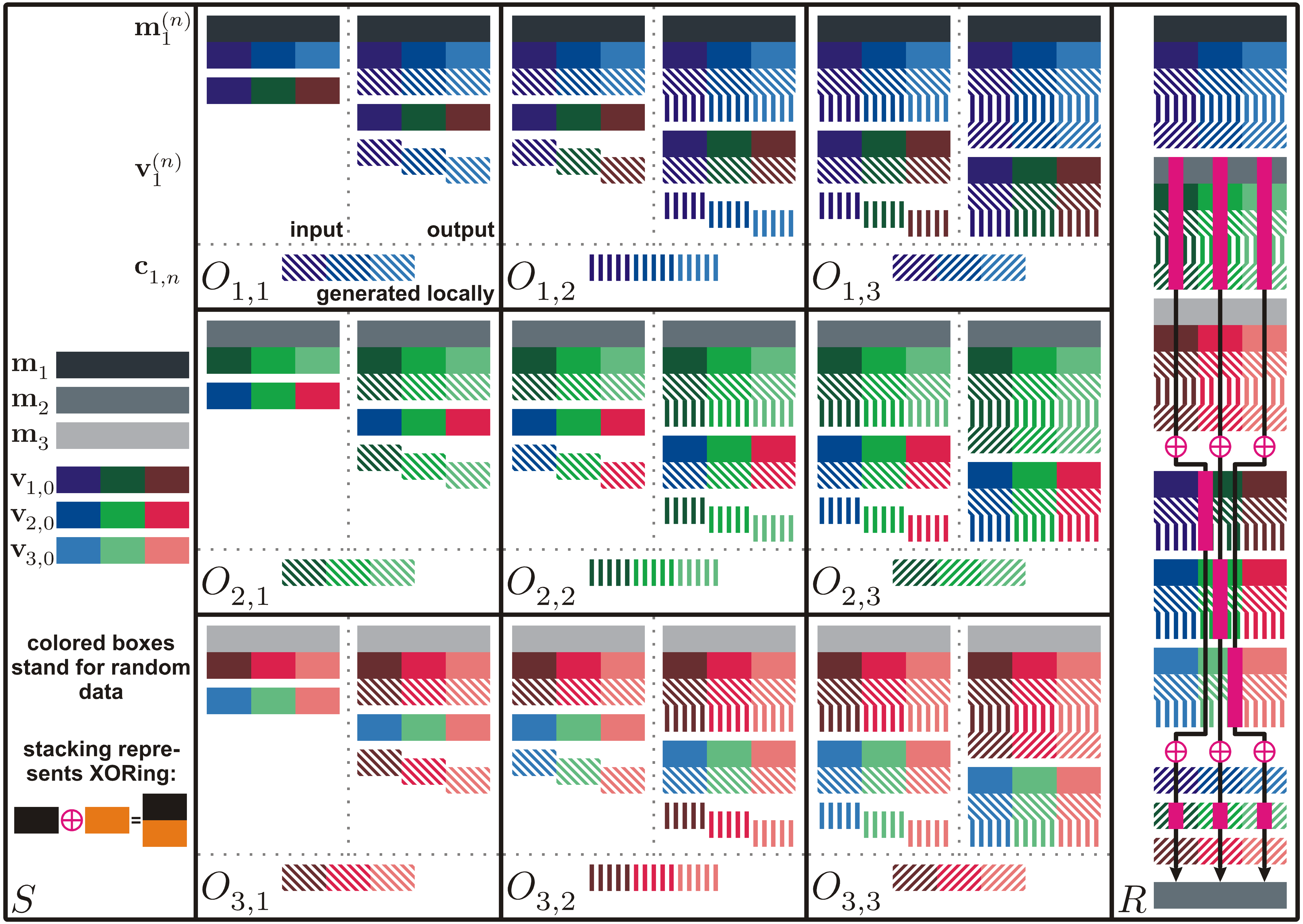
[13] R. Steinmetz, K. Wehrle, "Peer-to-Peer Systems and Applications",
Springer, ISBN 978-3-540-29192-3

[14] S. Katti, D. Katabi, K. Puchala, "Slicing the Onion: Anonymous Routing without PKI",
ACM HotNets, 2005.

[15] M. J. Freedman and R. Morris,
"Tarzan: A peer-to-peer anonymizing network layer",
Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), 2002.

[16] R. Dingledine, N. Mathewson, P. F. Syverson,
"Tor: The second generator onion router",
Proceedings of the 13th USENIX Security Symposium, pp. 303-320, 2004.

[17] D. Chaum,
"Untraceable electronic mail, return addresses, and digital pseudonyms",
Comm. of the ACM, vol.24, no. 11, pp. 84-90, 1981.


$\mathbf{m}_1^{(n)}$
$\mathbf{v}_1^{(n)}$
$\mathbf{c}_{1,n}$
input
output
generated locally
$O_{1,1}$
$O_{1,2}$
$O_{1,3}$
$O_{2,1}$
$O_{2,2}$
$O_{2,3}$
$O_{3,1}$
$O_{3,2}$
$O_{3,3}$
$\mathbf{m}_1$
$\mathbf{m}_2$
$\mathbf{m}_3$
$\mathbf{v}_{1,0}$
$\mathbf{v}_{2,0}$
$\mathbf{v}_{3,0}$
colored boxes stand for random data
stacking represents XORing:
$\oplus$
=
S
R