Churn Resilience in Network Coding-based Anonymous P2P System

Todorka Alexandrova
Waseda University
Tokyo, Japan
Email: alexandrova@aoni.waseda.jp

Gergely Huzsak
University of Electro-Communications
Tokyo, Japan
Email: gergely@appnet.is.uec.ac.jp

Hiroyoshi Morita
University of Electro-Communications
Tokyo, Japan
Email: morita@is.uec.ac.jp

***Abstract*—Churn resilience is an important issue for current Peer-to-Peer(P2P) networks, where peers dynamically and freely join and leave the system with or without warning. In this paper we focus on improving an existing anonymous peer-to-peer network design [8] by adding churn-resilience to it. Churn resilience is achieved by employing the properties of threshold secret sharing schemes. The paper describes the proposed method, evaluates the churn resilience in it and describes the future research steps to be taken.**

I. Introduction

Peer-to-Peer (P2P) networks have become quite popular in the recent years for their ease of use and powerful resource sharing scheme. Since the birth of P2P networks and of privacy concerns, research on the field of anonymous communication has been vivid. The last decade has witnessed major progress in the field of Anonymous Peer-to-Peer Communication (AP2P) Networks.

Anonymous communication refers to the communication between two participants when at least one of the participants is unaware of the identity of the other, and an attacker can not obtain the identity of at least one of the participants, and can not identify the contents of the communication exchanged between them. Starting from the first designs (Anonymizer [9], Napster [10], etc.) the evolution has been steady and yielded several working solution, where a popular class are overlays based on relaying (forwarding through intermediary nodes) like Chaumian networks [1], Onion Routing [4], Tor [6], Crowds [7], etc.

In general, there are two main stages in the realization of anonymous systems that are based on relaying. The first one is the anonymous path setup, which consists of choosing a set of relay nodes and appropriately arranging them between the sender and the receiver. The second stage is the anonymous delivery of data using the pre-arranged nodes.

However, for realizing anonymity, most of the proposed schemes (Onion Routing[4], Tor [6], etc.) depend heavily on public key infrastructure (PKI) or require a trusted central entities to store pre-shared keys (Crowds [7]) needed during the process of informing each node of its next hop in the path set up stage and while transmitting the data. The need for PKI makes the system less efficient and vulnerable to security breaches. Moreover it is hard to deploy key infrastructure in distributed environment, which is another reason to consider this as a possible drawback.

To solve the PKI requirement problem, Katti et al. [2], [3], proposed the *Information Slicing* technique, which realizes anonymous communication without using any PKI. Instead of creating an anonymous path as in the onion routing idea, the information slicing method first establishes the forwarding graph by sending each intermediate node its routing information (i. e., its next hop's IP address) in a confidential message sliced over multiple disjoint paths and then sends the actual data through this graph.

One of the more recent works [8] proposes a new anonymous communication protocol, called AC-ITNC. It is essentially a two-stage relaying method, where the first stage, related to the anonymous path setup, is realized by a novel and computationally efficient method, called *Information Transmission based on Network Coding (ITNC)*. The second data delivery stage is realized by onion routing like protocol. ITNC is a new information slicing and transmission technique based on multi-path network coding that relies purely on random number generation and Exclusive Or (XOR) operation-based Network Coding to share routing information with relay nodes over a mesh. It claims to have improved the conspiracy attack weaknesses of Katti et al. [2], [3]. In ITNC, instead of just transmitting the slices of the messages by the intermediate nodes as in [2], all the intermediate nodes on the forwarding path generate random vectors to further encode the received pieces. The paper [8] claims to have remarkable properties with respect to anonymity, while remaining computationally efficient, but it does not address the problem of *churn resilience* in the ITNC method and *relies heavily on the fact that every node is present, receives/forwards every packet to the destinations unaltered.*

Our definition of *churn* refers to the failure of a node to carry out its coding and/or forwarding duties, due to peers dynamically joining/leaving the network with or without warning, to network/hardware errors or to the purposeful malevolent action of attacker(s), ultimately resulting in message loss/corruption/duplication. Due to the fact that churn and attacks are inherent to any large and open P2P network, we believe that any design striving for stepping beyond the theoretical realm must deal with this aspects thoroughly.

Due to the fact that the number of nodes participating in the routing information distribution of the ITNC is large (at least quadratic), in this paper we focus mainly on the first stage of the AC-ITNC protocol and address the churn resilience

at this stage. We propose an extended churn resilient ITNC method, which is achieved by double-encoding using threshold secret sharing schemes [5]. The approach's churn resilience is evaluated and examples showing its validity are presented. Our proposal is to keep the known positive properties of the original system, while protecting it against churn and attacks.

II. Secret Sharing Schemes

Threshold secret sharing schemes were introduced by Shamir in 1979 [5]. A secret sharing scheme is called a (t, m) threshold secret sharing scheme if knowledge of any t or more shares makes the secret s computable and the knowledge of any $t-1$ or fewer shares leaves s completely undetermined (in the sense that all its possible values are equally likely).

Let $\mathcal{K}$ and $\mathcal{S}$ be the sets of secrets and shares, respectively. Suppose a dealer D selects a secret $s \in \mathcal{K}$ and distributes shares $s_i \in \mathcal{S}$ to each user $P_i \in \mathcal{P}$, where $\mathcal{P} \triangleq \{P_1, .., P_m\}$ and $D \notin \mathcal{P}$. Shamir's (t, m) threshold secret sharing scheme is described as follows [5]:

1. Let $\mathcal{K} = GF(q)$ and $\mathcal{S} = GF(q)$, where q is a prime power and $q \geq m + 1$.

2. D chooses elements $a_1, \ldots, a_{t-1} \in GF(q)$ independently and uniformly, and constructs the polynomial $f(x)$ $f(x) = s + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$.

3. D chooses m distinct points $x_i \in GF(q)$, $1 \leq i \leq m$. The values x_i are public.

4. D distributes shares $s_i = f(x_i)$ to users P_i, $1 \leq i \leq m$.

Using polynomial interpolation every t or more users can recover the secret s and any group of $t-1$ or less users obtains no information about the secret.

III. Proposed Churn Resilient ITNC Method

In this section we describe the proposed churn resilient ITNC method, that securely transmits information between two nodes without using cryptographic technique.

Basic Notations:

- $\mathbf{m}$ - a message $\mathbf{m}$ to be sent between S and R;
- m - number of paths in the forwarding mesh;
- n - number of nodes per path in the forwarding mesh;
- $O_{i,j}$ - the jth node on the ith path of the mesh;
- $\oplus$ - XOR operation;
- $\mathbf{a} \oplus \mathbf{b}$ - vector XOR operation, where $\mathbf{a} \oplus \mathbf{b} = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n)$ for binary vectors $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$;
- $(\mathbf{a} \odot \mathbf{b})$ - concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$.

Let S be the sender, that wants to send a message $\mathbf{m}$ to the receiver R. The message transmission between S and R is achieved securely, using the following steps.

A. Forwarding Mesh

First, S choses a set of $m \times n$ relay nodes and arranges them in a mesh (network) as shown in Figure 1, where in the mesh there are m disjoint paths with n nodes per path and $O_{i,j}$ stands for the jth node on the ith path of the mesh.

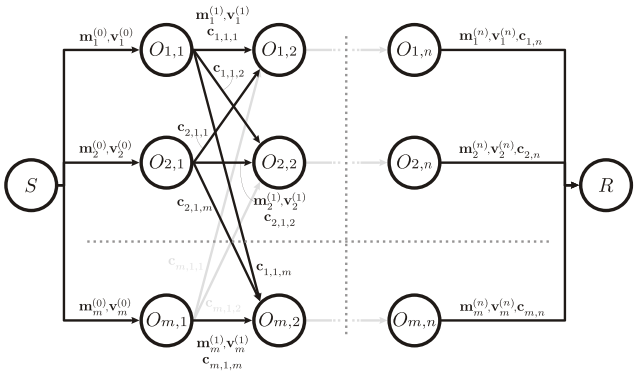

Fig. 1. Forwarding Mesh

B. Message Sharing at S

S performs double-encoding on the message $\mathbf{m}$ by applying twice threshold secret sharing schemes. The two steps of the double-encoding are described in **(i)** and **(ii)** as follows:

(i) *Step 1*: S shares the message $\mathbf{m}$ into m shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \ldots, \tilde{\mathbf{m}}_m$ using a (t, m) threshold secret sharing scheme as described in Section II. If $\mathbf{m}$ is a message with size a, i.e., $\mathbf{m} \in GF(2)^a$, then $\tilde{\mathbf{m}}_i \in GF(2)^a$, for $i = 1, \ldots, m$.

(ii) *Step 2*: S shares each of the shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \ldots, \tilde{\mathbf{m}}_m$ again using a (t, m) threshold secret sharing scheme, where $\tilde{\mathbf{m}}_i^1, \tilde{\mathbf{m}}_i^2, \ldots, \tilde{\mathbf{m}}_i^m$ stand for the shares of $\tilde{\mathbf{m}}_i$, for $i = 1, \ldots, m$. Then obviously $\tilde{\mathbf{m}}_i^j \in GF(2)^a$, for $j = 1, \ldots, m$.

Then, S forms the pseudo messages

$$\mathbf{m}_i = (\odot_{l=1}^{m} \tilde{\mathbf{m}}_i^l),$$

where $\mathbf{m}_i \in GF(2)^{am}$ and let $d = am$. These pseudo messages are transmitted in a special manner over the network mesh as shown in Figure 1. The information coding and transmission method of the pseudo messages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_m$ follows exactly the same ITNC procedure proposed in [8]. For the better understanding of the later churn resilience discussion, we describe the ITNC procedure [8] in the following subsections. However, more detailed description of the ITNC method can be found in [8].

C. Information Coding and Transmission at S

For each $\mathbf{m}_i \in GF(2)^d$, S chooses a random vector $\mathbf{c}_{i,0} \in GF(2)^d$ as a coding coefficient and computes

$$\mathbf{m}_i^{(0)} = \mathbf{c}_{i,0} \oplus \mathbf{m}_i, \text{for } i = 1, \ldots, m. \tag{1}$$

The coding coefficient $\mathbf{c}_{i,0}$ is also divided into m elements:

$$\mathbf{c}_{i,0} = (\odot_{l=1}^{m} \mathbf{c}_{i,0,l}),$$

where $\mathbf{c}_{i,0,l} \in GF(2)^a$, for $l = 1, \ldots, m$.
Thus, Eq. 1 can be also written as

$$\mathbf{m}_i^{(0)} = (\odot_{l=1}^{m} \mathbf{c}_{i,0,l}) \oplus (\odot_{l=1}^{m} \tilde{\mathbf{m}}_i^l). \tag{2}$$

Let $\mathbf{v}_i^{(0)}$ for some $1 \leq i \leq m$ be the following vector

$$\mathbf{v}_i^{(0)} = (\odot_{l=1}^{m} \mathbf{c}_{l,0,i}).$$

Then as shown in Figure 1, S forwards $\mathbf{m}_i^{(0)}$ and $\mathbf{v}_i^{(0)}$ to node $O_{i,1}$ along path i for $i = 1, \ldots, m$. Node $O_{i,1}$ can not recover $\mathbf{m}_i$ since it receives only a single piece $\mathbf{c}_{i,0,i}$ of $\mathbf{c}_{i,0}$.

D. Information Coding and Transmission at Intermediate Node

As described, node $O_{i,1}$, $i = 1, \ldots, m$ receives $\mathbf{m}_i^{(0)}$ and $\mathbf{v}_i^{(0)}$ from S. As shown in Figure 1, a node $O_{i,j}$ for $i = 1, \ldots, m$ and $j = 2, \ldots, n$ receives messages $\mathbf{m}_i^{(j-1)}$, $\mathbf{v}_i^{(j-1)}$ and $\mathbf{c}_{i,j-1,i}$ from $O_{i,j-1}$ and the coefficient elements $\mathbf{c}_{k,j-1,i}$ from nodes $O_{k,j-1}$ for $k = 1, 2, \ldots, m$ and $k \neq i$.

Then, there are basically three steps that each of the nodes $O_{i,j}$ performs:

(i) Chooses a random coefficient $\mathbf{c}_{i,j} \in GF(2)^d$ to encode the message $\mathbf{m}_i^{(j-1)}$

$$\mathbf{m}_i^{(j)} = \mathbf{c}_{i,j} \oplus \mathbf{m}_i^{(j-1)} = (\oplus_{r=0}^{j} \mathbf{c}_{i,r}) \oplus \mathbf{m}_i = \mathbb{O}_{l=1}^{m} \{\oplus_{r=0}^{j} \mathbf{c}_{i,r,l} \oplus \tilde{\mathbf{m}}_i^l\}. \quad (3)$$

(ii) Calculates $\mathbf{v}_i^{(j)}$ from the received messages:

For nodes $O_{i,1}$, $i = 1, \ldots, m$, $\mathbf{v}_i^{(1)} = \mathbf{v}_i^{(0)}$ as a special case. Node $O_{i,j}$ for $i = 1, \ldots, m$ and $j = 2, \ldots, n$ forms the vector $\mathbf{c}'_{j-1,i} = (\mathbb{O}_{l=1}^{m} \mathbf{c}_{l,j-1,i})$ and calculates

$$\mathbf{v}_i^{(j)} = \mathbf{c}'_{j-1,i} \oplus \mathbf{v}_i^{(j-1)} = (\mathbb{O}_{l=1}^{m} \{\oplus_{r=0}^{j-1} \mathbf{c}_{l,r,i}\}). \quad (4)$$

(iii) Similar to S, node $O_{i,j}$, $j < n$ also divides the coding coefficient $\mathbf{c}_{i,j}$ into m elements:

$$\mathbf{c}_{i,j} = (\mathbb{O}_{l=1}^{m} \mathbf{c}_{i,j,l}),$$

where $\mathbf{c}_{i,j,l} \in GF(2)^a$, for $l = 1, \ldots, m$. Then, node $O_{i,j}$ transmits the message $\{\mathbf{m}_i^{(j)}, \mathbf{v}_i^{(j)}, \mathbf{c}_{i,j,i}\}$ to next node on the ith path $O_{i,j+1}$ and elements $\mathbf{c}_{i,j,k}$ to nodes $O_{k,j+1}$, for $k = 1, 2, \ldots, m$ and $k \neq i$, respectively.

The last nodes $O_{i,n}$, $i = 1, 2, \ldots, m$, directly forward the messages $\{\mathbf{m}_i^{(n)}, \mathbf{v}_i^{(n)}, \mathbf{c}_{i,n}\}$ to the receiver R.

E. Decoding at the Receiver

In this section we show how the message $\mathbf{m}$ can be recovered at R from the messages obtained.

First we describe how the pseudo messages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_m$ are obtained and then we show how the original message $\mathbf{m}$ can be obtained from them.

It is obvious that R can decode the messages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_m$ from the messages it receives following exactly the same procedure given in [8]. Here we give a brief description of the decoding method and more detailed one can be found in [8]. R receives $\mathbf{c}_{i,n}$ and the messages

$$\mathbf{m}_i^{(n)} = \mathbf{c}_{i,n} \oplus \ldots \oplus \mathbf{c}_{i,1} \oplus \mathbf{c}_{i,0} \oplus \mathbf{m}_i,$$

$$\mathbf{v}_i^{(n)} = (\mathbb{O}_{l=1}^{m} \{\oplus_{r=0}^{n-1} \mathbf{c}_{l,r,i}\}), \text{ for } i = 1, 2, \ldots, m.$$

Then, $\mathbf{m}_i$, for $i = 1, \ldots, m$ can be computed as follows

$$(\mathbb{O}_{l=1}^{m} \{\oplus_{r=0}^{n-1} \mathbf{c}_{i,r,l}\}) \oplus \mathbf{c}_{i,n} \oplus \mathbf{m}_i^{(n)} = \oplus_{r=0}^{n} \mathbf{c}_{i,r} \oplus \mathbf{m}_i^{(n)} = \mathbf{m}_i. \quad (5)$$

From the pseudo message $\mathbf{m}_i$ messages $\tilde{\mathbf{m}}_i^1, \tilde{\mathbf{m}}_i^2, \ldots, \tilde{\mathbf{m}}_i^m$ are obtained, since $\mathbf{m}_i = (\tilde{\mathbf{m}}_i^1 | \tilde{\mathbf{m}}_i^2 | \ldots | \tilde{\mathbf{m}}_i^m)$. From $\tilde{\mathbf{m}}_i^1, \tilde{\mathbf{m}}_i^2, \ldots, \tilde{\mathbf{m}}_i^m$ using the (t, m) secret sharing scheme recovery technique described in Section II, message $\tilde{\mathbf{m}}_i$ can be obtained. Analogically $\tilde{\mathbf{m}}_i$ for all $i = 1, 2, \ldots, m$ can be computed as well. Then using again the (t, m) secret sharing scheme method the original message $\mathbf{m}$ can be recovered from shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \ldots, \tilde{\mathbf{m}}_m$.

Remark 1: As a strong advantage of the proposed system it is worth mentioning that not all the pseudo messages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_m$ are needed in order to recover the messages $\mathbf{m}$ in contrast to the original ITNC method [8], which utilizes only the (m, m) secret sharing scheme scenario. If at least t of these messages are obtained, then exploiting the features of the (t, m) threshold secret sharing schemes, the original message can be still easily recovered. Moreover, following the same reason, if at least t of the $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \ldots, \tilde{\mathbf{m}}_m$ messages are available, the original message can be recovered. This is the key point of the churn resilience proposal in this paper and will be discussed in more details in the following section.

IV. Churn Resilience in the Proposed Extended-ITNC System

In this section we present the churn resilience features of the proposed in Section III extended churn resilient ITNC method. As already mentioned, the original ITNC method [8] relies heavily on the fact that every node is present, receives/forwards every packet destinations unaltered and the recovery of the original message $\mathbf{m}$ is possible only in case all the messages from all nodes $O_{i,n}$, $i = 1, \ldots, m$ are received. However, using the scheme presented in this paper, the original message $\mathbf{m}$ can be recovered in a lossless manner even if churn is present in the system. The effective strength of the resilience is fully configurable by proper selection of the scheme's basic parameters.

For the purpose of simplicity, we first analyze the churn resilience of the proposal through examples.

A. Example Scenario

Here we present an example of the proposed churn resilient ITNC algorithm for $t = 2$, $m = 3$ and $n = 3$.

First, S shares the message $\mathbf{m}$ into three shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \tilde{\mathbf{m}}_3$ using a $(2, 3)$ threshold secret sharing scheme. Then, S shares the shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \tilde{\mathbf{m}}_3$ again using a $(2, 3)$ threshold secret sharing scheme, where $\tilde{\mathbf{m}}_i^1, \tilde{\mathbf{m}}_i^2, \tilde{\mathbf{m}}_i^3$ stand for the shares of $\tilde{\mathbf{m}}_i$, for $i = 1, 2, 3$ and forms the pseudo messages $\mathbf{m}_i = (\mathbb{O}_{l=1}^{3} \tilde{\mathbf{m}}_i^l)$. Then these messages are transmitted and encoded as described in Section III via the mesh in Figure 1.

The very same process is illustrated using an alternative representation in Figure 2, which gives more details about the coefficient $\mathbf{c}_{i,j}$ generation as well as the XOR computation of messages $\mathbf{v}_i^{(j)}$ and $\mathbf{m}_i^{(j)}$ at each node $O_{i,j}$.

The right side of Figure 2 shows the technique for recovering the pseudo message $\mathbf{m}_2$. The pink rectangles show the parts of the corresponding messages to be XOR-ed in order to recover $\mathbf{m}_2$. In a similar way the recovery of pseudo messages $\mathbf{m}_1$ and $\mathbf{m}_3$ can be represented.

However, as already stated, in the original ITNC [8] all the pseudo messages $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ are needed in order to recover

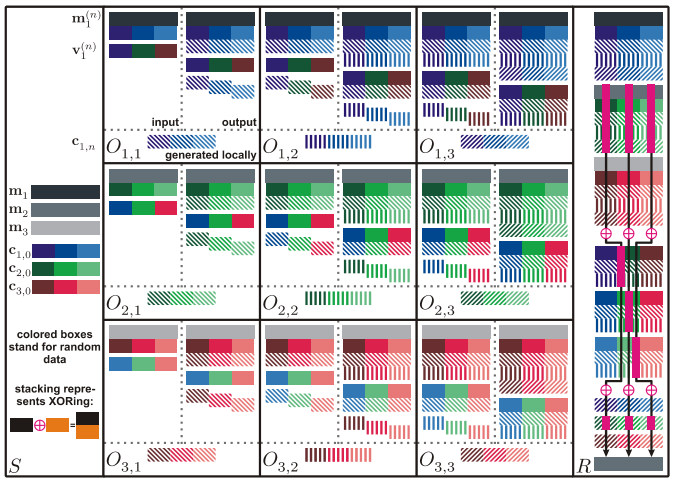

Fig. 2. Example's routing and message recovery

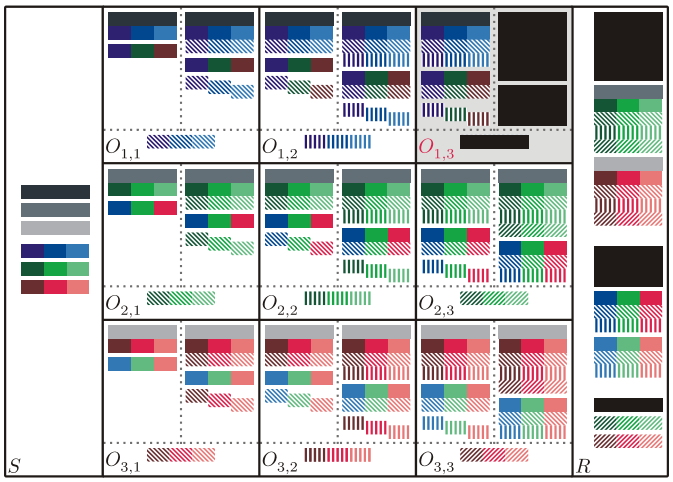

Fig. 3. Failure at node $O_{1,3}$

the original message $\mathbf{m}$. In the proposed extended ITNC if at least two of these messages are obtained, then exploiting the features of the $(2,3)$ threshold secret sharing schemes, the original message $\mathbf{m}$ can be still easily recovered. Moreover, following the same reason, if at least 2 of the $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \tilde{\mathbf{m}}_3$ messages are available, the original message can be obtained. Next we show how churn in the example can be handled by the proposed extended ITNC.

The recovery of $\mathbf{m}$ at R will be presented for three different cases. For all the cases for messages $\mathbf{m}_i^{(3)}$, $i = 1, 2, 3$, from Eq. (3) we have $\mathbf{m}_i^{(3)} = (\mathbb{O}_{l=1}^{3}\{\oplus_{r=0}^{3}\mathbf{c}_{i,r,l} \oplus \tilde{\mathbf{m}}_i^l\})$.

B. Case 1: Failure at node $O_{1,3}$

Here we show how $\mathbf{m}$ is recovered by R if a failure at node $O_{1,3}$ occurs. In such case, the unaltered messages R receives are $\mathbf{c}_{2,3}$, $\mathbf{c}_{3,3}$, $\mathbf{m}_2^{(3)}$, $\mathbf{m}_3^{(3)}$ and $\mathbf{v}_2^{(3)} = (\mathbb{O}_{l=1}^{3}\{\oplus_{r=0}^{2}\mathbf{c}_{l,r,2}\})$, $\mathbf{v}_3^{(3)} = (\mathbb{O}_{l=1}^{3}\{\oplus_{r=0}^{2}\mathbf{c}_{l,r,3}\})$.

R does not receive all parts of $\mathbf{m}_1^{(3)}$, $\mathbf{v}_1^{(3)}$ and $\mathbf{c}_{1,3}$, as shown in Figure 3, where the violated messages are shown in black. Then R calculates similarly to Eq. (6):

$(\mathbb{O}_{l=2}^{3}\{\oplus_{r=0}^{2}\mathbf{c}_{2,r,l}\})\oplus(\mathbb{O}_{l=2}^{3}\{\oplus_{r=0}^{3}\mathbf{c}_{2,r,l}\oplus\tilde{\mathbf{m}}_2^l\})\oplus(\mathbb{O}_{l=2}^{3}\mathbf{c}_{2,3,l})=(\mathbb{O}_{l=2}^{3}\tilde{\mathbf{m}}_2^l)$.

Thus, R is able to recover messages $\tilde{\mathbf{m}}_2^2, \tilde{\mathbf{m}}_2^3$. Analogically messages $\tilde{\mathbf{m}}_3^2, \tilde{\mathbf{m}}_3^3$ can be recovered. Using the $(2,3)$ secret sharing schemes $\tilde{\mathbf{m}}_2$ can be obtained from $\tilde{\mathbf{m}}_2^2, \tilde{\mathbf{m}}_2^3$ and $\tilde{\mathbf{m}}_3$ can be obtained from $\tilde{\mathbf{m}}_3^2, \tilde{\mathbf{m}}_3^3$. Then, using again $(2,3)$ secret sharing schemes for the shares $\tilde{\mathbf{m}}_2, \tilde{\mathbf{m}}_3$, the original message $\mathbf{m}$ is recovered. Analogically the situation for failures at nodes $O_{2,3}$ and $O_{3,3}$ can be handled.

C. Case 2: Failure at node $O_{2,2}$

Here we show how $\mathbf{m}$ is recovered by R if a failure at node $O_{2,2}$ occurs. In such a case, the unaltered messages R receives are $\mathbf{c}_{1,3}$, $\mathbf{c}_{3,3}$, $\mathbf{m}_1^{(3)}$, $\mathbf{m}_3^{(3)}$ and parts of messages $\mathbf{v}_1^{(3)}$ and $\mathbf{v}_3^{(3)}$, denoted by $\tilde{\mathbf{v}}_1^{(3)}$ and $\tilde{\mathbf{v}}_3^{(3)}$, respectively, where

$\tilde{\mathbf{v}}_1^{(3)} = (\mathbb{O}_{l=\{1,3\}}\{\oplus_{r=0}^{2}\mathbf{c}_{l,r,1}\})$, $\tilde{\mathbf{v}}_3^{(3)} = (\mathbb{O}_{l=\{1,3\}}\{\oplus_{r=0}^{2}\mathbf{c}_{l,r,3}\})$.

Then R calculates similarly to Eq. (6):

$(\mathbb{O}_{l=\{1,3\}}\{\oplus_{r=0}^{2}\mathbf{c}_{1,r,l}\}) \oplus (\mathbb{O}_{l=\{1,3\}}\{\oplus_{r=0}^{3}\mathbf{c}_{1,r,l} \oplus \tilde{\mathbf{m}}_1^l\}) \oplus (\mathbb{O}_{l=\{1,3\}}\mathbf{c}_{1,3,l}) = (\mathbb{O}_{l=\{1,3\}}\tilde{\mathbf{m}}_1^l)$.

Thus, R is able to recover messages $\tilde{\mathbf{m}}_1^1, \tilde{\mathbf{m}}_1^3$. Analogically messages $\tilde{\mathbf{m}}_3^1, \tilde{\mathbf{m}}_3^3$ can be recovered. Then using the $(2,3)$ secret sharing schemes $\tilde{\mathbf{m}}_1$ can be obtained from $\tilde{\mathbf{m}}_1^1, \tilde{\mathbf{m}}_1^3$ and $\tilde{\mathbf{m}}_3$ can be obtained from $\tilde{\mathbf{m}}_3^1, \tilde{\mathbf{m}}_3^3$. Then using again $(2,3)$ secret sharing schemes for the shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_3$, the original message $\mathbf{m}$ is recovered. Analogically the situation for failure at nodes $O_{1,2}$ and $O_{3,2}$ can be handled.

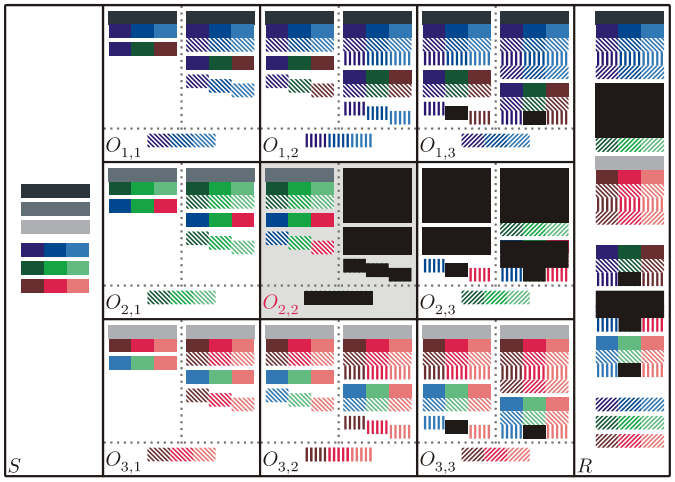

Fig. 4. Failure at node $O_{2,2}$

D. Case 3: Failure at node $O_{3,1}$

Here we show how $\mathbf{m}$ is recovered by R if a failure at node $O_{3,1}$ occurs. In such a case, the unaltered messages R receives are $\mathbf{c}_{1,3}$, $\mathbf{c}_{2,3}$, $\mathbf{m}_1^{(3)}$, $\mathbf{m}_2^{(3)}$ and parts of messages $\mathbf{v}_1^{(3)}$ and $\mathbf{v}_2^{(3)}$, denoted by $\tilde{\mathbf{v}}_1^{(3)}$ and $\tilde{\mathbf{v}}_2^{(3)}$, respectively, where $\tilde{\mathbf{v}}_1^{(3)} = (\mathbb{O}_{l=1}^{2}\{\oplus_{r=0}^{2}\mathbf{c}_{l,r,1}\})$, $\tilde{\mathbf{v}}_2^{(3)} = (\mathbb{O}_{l=1}^{2}\{\oplus_{r=0}^{2}\mathbf{c}_{l,r,2}\})$.

Then R calculates similarly to Eq. (6):

$(\mathbb{O}_{l=1}^{2}\{\oplus_{r=0}^{2}\mathbf{c}_{1,r,l}\})\oplus(\mathbb{O}_{l=1}^{2}\{\oplus_{r=0}^{3}\mathbf{c}_{1,r,l}\oplus\tilde{\mathbf{m}}_1^l\})\oplus(\mathbb{O}_{l=1}^{2}\mathbf{c}_{1,3,l})=(\mathbb{O}_{l=1}^{2}\tilde{\mathbf{m}}_1^l)$.

Thus, R is able to recover messages $\tilde{\mathbf{m}}_1^1, \tilde{\mathbf{m}}_1^2$. Analogically messages $\tilde{\mathbf{m}}_2^1, \tilde{\mathbf{m}}_2^2$ can be recovered. Then using the $(2,3)$ secret sharing schemes $\tilde{\mathbf{m}}_1$ can be obtained from $\tilde{\mathbf{m}}_1^1, \tilde{\mathbf{m}}_1^2$ and

$\tilde{\mathbf{m}}_2$ can be obtained from $\tilde{\mathbf{m}}_2^1, \tilde{\mathbf{m}}_2^2$. Then using again $(2,3)$ secret sharing schemes for the shares $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2$, the original message $\mathbf{m}$ is recovered. Analogically the situation for failure at nodes $O_{1,1}$ and $O_{2,1}$ can be handled.

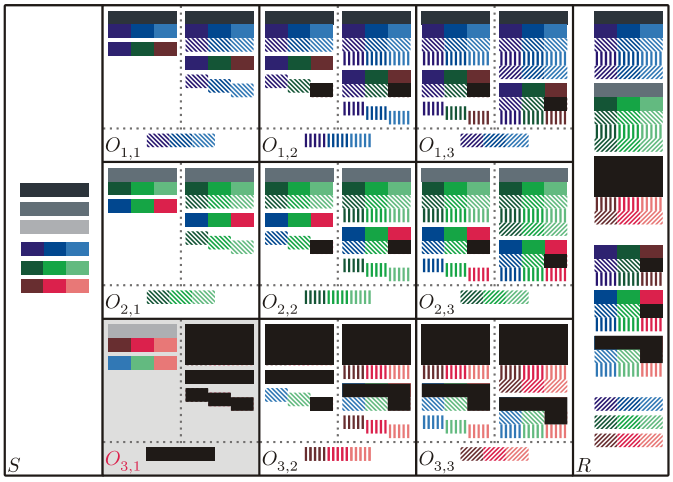

Fig. 5. Failure at node $O_{3,1}$

Theorem 1 given below generalizes the churn resilience analysis in this section.

Theorem 1. *Message* $\mathbf{m}$ *can be recovered as long as the number of paths where any churn is present is at most* $m-t$.

Proof: Assume that at least one churn has occurred on paths from 1 to $m-t$ and there is no churn on the other t paths left. In such a case, for sure R receives fully the messages $\mathbf{m}_{m-t+1}^{(n)}$, $\mathbf{m}_{m-t+2}^{(n)}$, ..., $\mathbf{m}_m^{(n)}$ and $\mathbf{c}_{m-t+1,n}$, $\mathbf{c}_{m-t+2,n}$, ..., $\mathbf{c}_{m,n}$, where

$$\mathbf{m}_i^{(n)} = (\mathbb{O}_{l=1}^{m}\{\oplus_{r=0}^{n}\mathbf{c}_{i,r,l} \oplus \tilde{\mathbf{m}}_i^l\}), \text{for } i = m-t+1, \ldots, m. \quad (6)$$

Due to churn some parts of the received messages $\mathbf{v}_{m-t+1}^{(n)}$, $\mathbf{v}_{m-t+2}^{(n)}$, ..., $\mathbf{v}_m^{(n)}$ are missing, changed or modified. However, since there are still t paths in the mesh without any churn occurring on them, there are parts of them, denoted by $\tilde{\mathbf{v}}^{(n)}$, that has been unaltered, where

$$\tilde{\mathbf{v}}_i^{(n)} = (\mathbb{O}_{l=m-t+1}^{m}\{\oplus_{r=0}^{n-1}\mathbf{c}_{l,r,i}\}), \text{for } i = m-t+1, \ldots, m. \quad (7)$$

Then from the messages received correctly, R calculates similarly to Eq. (6):
$(\mathbb{O}_{l=m-t+1}^{m}\{\oplus_{r=0}^{n-1}\mathbf{c}_{m-t+1,r,l}\}) \oplus (\mathbb{O}_{l=m-t+1}^{m}\{\oplus_{r=0}^{n}\mathbf{c}_{m-t+1,r,l} \oplus \tilde{\mathbf{m}}_{m-t+1}^l\}) \oplus (\mathbb{O}_{l=m-t+1}^{m}\mathbf{c}_{m-t+1,n,l}) = (\mathbb{O}_{l=m-t+1}^{m}\tilde{\mathbf{m}}_{m-t+1}^l)$.

Thus, R is able to recover the t messages $\tilde{\mathbf{m}}_{m-t+1}^{m-t+1}, \tilde{\mathbf{m}}_{m-t+1}^{m-t+2}, \ldots, \tilde{\mathbf{m}}_{m-t+1}^{m}$. Then using the (t,m) threshold secret sharing schemes the message $\tilde{\mathbf{m}}_{m-t+1}$ can be obtained. Analogically, messages $\tilde{\mathbf{m}}_{m-t+2}, \ldots, \tilde{\mathbf{m}}_m$ can be recovered. Finally, using again (t,m) threshold secret sharing schemes the original message $\mathbf{m}$ can be computed from the t shares $\tilde{\mathbf{m}}_{m-t+1}, \tilde{\mathbf{m}}_{m-t+2}, \ldots, \tilde{\mathbf{m}}_m$. ■

E. Churn Analysis

We show the churn resilience efficiency of our approach compared to the ITNC method via analysis. Let the probability of node churn be p. Then the probability of path success is $P(path\,success) = (1-p)^n$. According to Theorem 1, the probability of the scheme succeeding is the probability of having at least t succeeding paths:

$$P(success) = \sum_{i=t}^{m} \binom{m}{i} (1-p)^{ni} (1-(1-p)^n)^{m-i}. \quad (8)$$

In the ITNC scheme on the other hand, no churn is allowed:

$$P(success) = (1-p)^{nm}. \quad (9)$$

Remark 2: The proposed method does not only add churn resilience to the system but also prevents adversaries from recovering the messages intended for R. Even if at most $m-t$ malicious nodes per stage exist, they are not able to obtain any idea about the original message $\mathbf{m}$ from all the data they have. Moreover, the current churn resilience scheme can be easily extended and used for tempering resilience as well.

Remark 3: Using (t,m) threshold secret sharing schemes in the two steps of the double-encoding as described in II.B will lead to the quadratic expansion of the size of the messages traversing the mesh. To overcome this limitation and improve the efficiency of the proposed system *ramp secret sharing schemes* can be easily applied in the second step of the double-encoding, instead of the threshold secret sharing schemes.

V. Conclusion and Future Work

In this paper an extended churn resilient ITNC method, exploiting the properties of double-encoding and threshold secret sharing schemes, have been proposed. Our goal is to achieve a design, that could become a real-life implementation serving the purpose of robust anonymous P2P communication withstanding wide range of attacks and failures. In order to achieve better efficiency, as a next step - instead of the secret sharing schemes - we will consider employing more optimal techniques (erasure codes, such as RC code etc.) for the second step of the message encoding.

We have also considered establishing bounds on attacker's chances of compromising anonymity to be able to compare our scheme's properties to those of the original design.

References

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

[2] S. Katti, J. Cohen, and D. Katabi "Information Slicing: Anonymity Using Unreliable Overlays," *Proc. of 4th USENIX Symposium on Networked Systems Design and Implementation*, pp.43-56, 2007.

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

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

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

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

[7] 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.

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

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

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