Efficient Unobservability against Strong Adversaries

BIU Dept. of CS, Network Security Group Technical Report 13-01 (Feb. 2013)

Nethanel Gelernter
Department of Computer Science
Bar Ilan University
Email: nethanel.gelernter@gmail.com

Amir Herzberg
Department of Computer Science
Bar Ilan University
Email: amir.herzberg@gmail.com

***Abstract*—We present Decentralized Unobservable Reporting Protocol (DURP), a low-latency, low-overhead, decentralized protocol for anonymous, unobservable communication to untrusted recipient. DURP ensures unobservability against global eavesdropper (and malicious participants), in contrast to known low-latency protocols such as Tor. DURP design is a modular combination of a simple queuing module, ensuring fixed rates for several events, together with an anonymization module, which can simply use existing anonymous protocols, specifically, either Onion-Routing (DURP^{OR}) or Crowds (DURP^{Crowds}).**

For length restrictions, we focus on network properties of DURP, e.g., latency. We found, using analysis backed by simulations, that DURP ensures reasonable latency and overhead. Our analysis allows configuring DURP for best trade-off of network properties.

I. Introduction

Anonymous communication is both important and challenging, motivating an abundance of research and systems [1]. However, significant challenges remain. In particular, solutions that ensure complete unobservable communication, like DC-net [2], have unacceptable overhead; practical solutions, like Tor, fail against eavesdroppers. Our goal is to develop practical protocols ensuring unobservability and anonymity. Practicality implies reasonable efficiency, robustness to failures and disruptions, and preferably a decentralized design.

In this paper, we focus on protocols for communication to a single destination. We believe it would help to consider a motivating scenario, which we refer to as the *shy crypto-students*.

Consider a class in Cryptography and Privacy, where students are shy to ask questions, lest they appear dumb. The professor wants to allow students to ask in an *unobservable* manner, i.e., without exposing the fact they asked (in fact, fellow students should not know *anyone* asked).

To meet this goal, she asks that every student, will write questions on a note, placed inside a sealed envelop, so that only she can read. Student can write on the note 'no question' if they have no question. If every student gives such envelops with the same rate as others (periodic or using same distribution), other students cannot know if anybody asked.

However, students are even concerned with the professor knowing they asked a dumb question. Namely, sender-anonymity from the professor is also needed. To solve this, with reasonable efficiency, the professor wants to use a variant of anonymity protocols such as Crowds [3] or Onion-Routing [4], namely, each student will handle envelopes of other students, as well as their own envelopes. For example, using the Crowds approach, each user forwards to another random student with probability P_f, and otherwise, with probability $1 - P_f$, gives the envelop to the lecturer. (The Onion-Routing approach also works fine.)

However, to prevent timing-based identification, students do not forward envelops immediately (as in Crowds and Onion-Routing). Instead, each student keeps a (hidden) pile of envelopes in a drawer, received from others or written by himself. The idea is for this pile to usually contain few envelops, so that when sending a random envelop from the pile, on each minute (round), neither the professor nor other students observing will know if the envelope sent is from the user or only forwarded from a different student; if the pile is empty, then the student simply sends an empty envelope. On the other hand, piles should not be too large, so that questions get to the lecturer on timely basis, while still discussing same topic.

To ensure piles are filled and depleted in a reasonable manner, each student adds a new envelop in a fixed rate - say, every minute, with a fixed probability $\frac{1}{\rho}$ for some $\rho > 1$. When adding an envelope, if the student has no question to ask, he simply adds an empty envelope.

This protocol achieves anonymity against the destination, with unobservability against strong attacker. Namely, the attacker may control multiple peers and eavesdrop on all communication. On the other hand, the protocol has reasonable communication and latency overheads. Furthermore, it is *robust*, i.e., a non-cooperating student cannot disrupt communication (or foil anonymity).

We note that the only known protocol that also achieves unobservability against a global eavesdropper, and some malicious participants, is the DC-net (Dining-Cryptographers) protocol of [2]; follow-up works improve this protocol, mainly by adding robustness. However, DC-net has very high communication overhead, since every bit requires each peer to send a bit, i.e., factor of $\mathcal{O}(n)$; its robust extensions have even higher overhead.

In this work, we study the simple protocol sketched above, with few necessary details added; we also study an optimized variant. This submission, due to length restrictions, focuses on the network properties (communication and latency overheads, sizes of buffers (piles), etc.); the (non-trivial) definitions and analysis of privacy properties appears in Appendices B C D.

A. Related Works: Anonymous Communication in Decentralized Networks

Several works present decentralized protocols for anonymous/unobservable communication. These solutions fail to satisfy at least one of our goals; specifically:

1) Efficiency: Chaum's DC-net [2], ensures unobservability under private channels like in our model; however, in the DC-net, the minimal ratio between real messages bits and dummy ones is $\mathcal{O}(n)$, when n is the number of the peers. Additional bandwidth is needed to deal with collisions [5]. In our solution, the minimal ratio is some constant (see more in Section IV).

2) Unobservability against eavesdroppers: Many solutions fail to ensure unobservability. Known protocols like Crowds [3], Onion Routing [4], TOR [6], hordes [7], MorphMix [8] and others, provide anonymity without using cover traffic; hence, they are not unobservable, even against local eavesdropper. In a network with other destinations for other purposes, and under the assumption of indistinguishability between the different message types, a local eavesdropper might not know whether a report or some dummy message was sent. However, a global eavesdropper can track all the messages and break anonymity.

3) Anonymity: Few works, most notably buses-based unobservability by Beimel and Dolev [9], achieve unobservability but not anonymity. (Beimel and Dolev sketch how to add anonymity, but with probability for message loss.)

4) Unobservability against strong attacker: Tarzan [10] is a peer to peer solution which use cover (dummy) traffic. Herbivore [11] is a peer to peer anonymity protocol based on Chaum's DC-net [2], that improves the DC-net efficiency. Both the protocols ensures unobservability against eavesdroppers, but many malicious peers can break the unobservability. Notice that the mere use of constant rate may not suffice to ensure unobservability;

B. Our contribution

We present and analyze the network properties of DURP, a decentralized modular protocol ('framework') that ensures unobservable reporting. DURP has a separate module which can be instantiated using different anonymity protocols, specifically, Crowds [3] and Onion Routing [4].

DURP is decentralized, without any single point of failure; this is important for reliability, security (e.g., against "legal attacks"), and for load balancing (required for efficient scalability). Unlike previous decentralized solutions (see above), DURP ensures unobservability against strong attackers, while maintaining efficiency. DURP is also easy to implement and deploy, reusing existing anonymity protocols modularly.

Another contribution of this work is precise probabilistic analysis, that predicts networking properties such as delays. The analysis can help in fine-tuning DURP, and for anomalies detection.

Organization of rest of paper. In section 2, we present the model and how we evaluate DURP's anonymity. In section 3 we introduce DURP and analyze its network properties. In section 4, we discuss DURP's performance, and present simulation results of two instantiations of DURP: using the Crowds protocol and using the Onion Routing protocol. We also compare the simulation results to the analysis of the previous section (with excellent match). In section 5 we discuss the relationship between DURP's network properties and security and anonymity concerns, and in section 6 we offer some improvements, analyze some of them, and compare the analysis with simulation results. In the last section we conclude our work and discuss future work.

II. Model and Requirements

A. Communication and Setup Model

For simplicity, we assume synchronous system, i.e. all peers operate in synchronous rounds and all the messages sent in one round, are received in the next round. We assume private channels between each pair of participants, e.g., using symmetric cryptography. We also assume that all the participants have the destination public key. Another assumption we make is that all the messages that need to be sent are of the same length.

We consider a network of $n-1$ peers and one destination, d, where every two participants can communicate. The peers want to send *anonymous* reports to the destination, such that *only the destination* can detect whether a report was sent or not.

B. Anonymity and Unobservability Definitions

We prove that using DURP we can ensure unobservability against any combination of eavesdroppers and malicious (active) peers, and anonymity against the destination. The proofs are based on a precise, indistinguishability based definition of such anonymity notions, extending the elegant definitions of [12]. The definitions in [12] deal only with a passive global eavesdropper; we extended them, to also consider malicious destination, as well as combinations of malicious peers and eavesdroppers.

The definitions cover multiple notions; in Theorem 26 we prove that DURP ensures unobservability against any combination of eavesdroppers and malicious (active) peers (Comp-UO-anon), and in Theorem 2 we prove that using DURP we also ensure strong source-anonymity against the destination (Comp-SA*-anon). Due to length restrictions, the definitions are presented in Appendices B C.

III. Decentralized Unobservable Reporting Protocol (DURP^{Π})

In this section we present and analyze DURP, a modular protocol that produces an unobservable reporting protocol DURP^{Π}, given a 'base' anonymous-communication protocol Π like Onion Routing [4] or Crowds [3]. We present an instantiation of DURP with the Onion Routing (OR) as a base protocol. The unobservability depends on both the base protocol Π and the network model; DURP^{Π} does not guarantee unobservability for every Π.

A. Operation of DURP

A DURP's peer (potential sender) is initialized with the security parameter k, the destination's public key pk_d, and ρ.

Every peer running DURP maintains a data-structure named *Set*. *Set* contains all the messages that the peer has to send to other participants (the other peers and the destination). The messages in *Set* are either received from other peers (messages that the peer has to relay), or inserted to *Set* from the application by the protocol (and their originator is the peer).

At every round, DURP$^{\Pi}$ sends exactly one message from *Set* (or a special *dummy* message, when *Set* is empty; The creation and the process of the messages is done by the base protocol Π (e.g., in Π=OR, when a message is created, the plaintext is wrapped as an onion [4]).

DURP maintains a fixed sending rate from the application at each peer to the destination. Namely, on every iteration, each peer adds a message from the application level to its *Set* with probability $\frac{1}{\rho}$. Notice that the application must provide a message; obviously, some signaling message can be sent to signal lack of real content, this should not be confused with the *dummy* message sent by DURP itself when *Set* is empty (see above). New messages from the application and dummy messages, are encrypted by DURP's public key CPA secure encryption scheme (see [13]), denoted by $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, with the (only) destination's key.

The pseudocode of DURP$^{\Pi}$ appears in Alg 1. An example to the base protocol Π as the Onion Routing protocol is brought in Alg 2.

Algorithm 1 DURP$^{\Pi}$. The Initialize, CreateMessage and ReceiveMessage methods are done by Π. In line 12 (c, p) is chosen uniformly.

1: **Initialization**(k, pk_d, ρ)**:**
2: $Set = \emptyset$
3: Π.**Initialize**(k)

4: **Every round do:**
5: **With** probability $\frac{1}{\rho}$ **do**
6: Get message m from the application
7: *Set*.add(Π.**CreateMessage**($\mathcal{E}_{pk_d}(m), d$))
8: **end do**
9: **if** *Set*.isEmpty() **then**
10: $(c, p) \leftarrow$ (Π.**CreateMessage**($\mathcal{E}_{pk_d}(dummy), d$))
11: **else**
12: $(c, p) \xleftarrow{r}$ *Set*
13: **end if**
14: Send c to p

15: **OnReceivingMessage**(c)**:**
16: *Set*.add(Π.**ReceiveMessage**(c))

B. DURP^{OR}'s Anonymity and Unobservability

1) Unobservability against eavesdroppers and malicious peers:

Theorem 1: (Informal) Let ψ be the public-key encryption scheme used by DURP^{OR}. If ψ is CPA-secure, then DURP^{OR} is Comp-**UO**-anonymous against any combination of eavesdroppers and malicious peers.

In Appendix D we give the formal theorem, and prove it.

2) Anonymity against the destination: DURP^{OR} does not ensure Comp-**SA***-anonymity against the destination. This happens because in iterations when the destination gets a message from the peer, it learns that in these iterations, the peer did not send any new message (the first hop of a message). The above version of DURP^{OR} provides anonymity, but not in the strong manner we want.

Algorithm 2 The Onion Routing protocol methods. $l > 1$ is the number of the layers (intermediate peers in a path) in the protocol.

1: OR.**Initialize**(k)**:**
2: $(pk, sk) \xleftarrow{\$} \mathcal{K}(k)$
3: Publish pk

4: OR.**CreateMessage**(m, d)**:**
5: $(p_1, p_2, ..., p_l) \xleftarrow{r}$ ChoosePath(l)
6: $(c, p) \leftarrow (m, d)$
7: **for** $i = l$ **to** 1 **do**
8: $(c, p) \leftarrow (\mathcal{E}_{pk_i}(c, p), p_i)$
9: **end for**
10: **return** (c, p)

11: OR.**ReceiveMessage**(c)**:**
12: **return** $(c, p) \leftarrow \mathcal{D}_{sk}(c)$

But, given the fact that DURP^{OR} ensures unobservability, we can use DURP^{OR} to achieve also anonymity against the destination as follows: We run DURP^{OR}, but instead of sending to the destination, a random peer is chosen. Every peer that receives a message as the destination, keeps the message in a separate queue. Every X iterations, we stop DURP^{OR}, and for one iteration every peer sends Y messages to the destination from the new queue; if a peer does not have enough messages to send the destination, it sends dummy messages. We denote this version of the DURP by DURP$^{\Pi}_{dest}$.

Theorem 2: (Informal) DURP$^{OR}_{dest}$ is Comp-**SA***-anonymous against malicious destination.

Theorem 26 and its proof hold also for DURP$^{OR}_{dest}$. We will discuss DURP$^{\Pi}_{dest}$ and prove Theorem 2 in the full version of the paper.

C. Notations

We next present few notations used in the analysis.

Iterations-counter of a message is the counter of iterations from the first iteration of the message in the network. This counter increases by one every iteration and stops when the message reaches its destination.

IC is the average *iterations-counter* of messages that reached their destinations.

The *hops-counter* of a message m, is the number of 'send' (Alg 1, line 14) events m passes.

We denote by $|\mathrm{C}reateMessage_{rnd}|$, the average number of 'CreateMessage' events during the round rnd. Similarly, we denote $|\mathrm{S}end_{rnd}|$ the average number of 'Send' events. The average hops-counter is denoted H, and defined as follows:

$$H \equiv \lim_{t\to\infty} \frac{\sum_{rnd=1}^{t} |\mathrm{S}end_{rnd}|}{\sum_{rnd=1}^{t} |\mathrm{C}reateMessage_{rnd}|}$$

H_i is the fraction of messages that are sent in iteration, and they have *hops-counter*=i, from the number of messages that are sent every iteration (on average).

ρ is the sending rate, i.e., every iteration, every peer has to add a message with probability $\frac{1}{\rho}$.

N is the number of the peers in the network; N_i is the average number of peers whose sets are of size i, when 'average' is over the entire execution. $N = \sum N_i$.

s_i is the average fraction of the *Set*s of size i, i.e., $s_i = \frac{N_i}{N}$, hence $\{s_i\}_{i=0}^{\infty}$ is the distribution of the *Set*s' sizes.

S is the average number of messages in the network. We denote by $|\mathrm{D}elivered_{rnd}|$ the number of messages that reached their destination in round rnd: $\lim_{t\to\infty} \sum_{rnd=1}^{t} |\mathrm{C}reateMessage_{rnd}| - |\mathrm{D}elivered_{rnd}|$.

D. Base Protocol (Π) Properties

For assuring the network properties, Π must meet two requirements:

1) Constant H: The bases protocol must have constant $H \geq 1$. Additionally, we demand that the *hops-counter* of every message when it reaches its destination (the final *hops-counter*), is taken independently from some probabilistic distribution over $\mathbb{N}$. For example: in dynamic routing protocol like Crowds [3], messages might reach their destination with different *hops-counter* values, but these values are determined by geometric distribution, and H is constant and defined to be $1 + \frac{1}{1-P_f}$, when P_f is the forwarding probability.

2) Relays are chosen uniformly: If the base protocol, Π, uses other peers as relays, these peers should be chosen uniformly. This requirement is necessary also for the anonymity.

E. Network properties

We now provide proof sketches for some network properties of DURP; the properties are affected by general attributes like H, and therefore the proofs are relevant for different base protocols.

A network running DURP$^{\Pi}$ is **stable**, if S is bounded and converging; the following lemma shows that the network is stable, provided that $\rho > H$; we assume this holds in the entire analysis. The proof of this lemma (and most others) appears in Appendix A.

Lemma 3: A network under DURP$^{\Pi}$ is *stable* when $\rho - H > 0$.

Lemma 4: The mean number of outgoing messages (messages that arrive to the destination) per iteration equals to the mean number of new messages per iteration, and both are $\frac{N}{H}$.

Proof: (sketch) If the mean of the new messages is greater, then S won't converge. On the other hand, the mean of the new messages cannot be less than the outgoing messages, because every outgoing message was created sometime. ■

Lemma 5: The proportion between the application messages and the dummy messages is $\frac{H}{\rho - H}$.

Proof: Over t iterations, N peers send messages tN times. Each message is sent, on average, H times until it reaches the destination. The average number of application messages during t iterations is: $\frac{tN}{\rho}$. The total number of 'send' events of application messages is therefore $\frac{HtN}{\rho}$. So, the dummy messages take $tN - \frac{HtN}{\rho}$ 'send' events. Because the mean of dummy message's *hops-counter* is the same as application message, the average number of dummy messages is $(tN - \frac{HtN}{\rho})\frac{1}{H}$. Division between the values gives the above result. ■

Lemma 6: $s_0 = \frac{\rho - H}{H(\rho - 1)}$.

Proof: According to the previous lemma, the average number of the dummy messages added every iteration is $\frac{N}{\rho} \cdot \frac{\rho - H}{H}$. By the protocol, dummy messages are created only in empty *Set*s. Before the creation of dummy message, there is a chance of $\frac{1}{\rho}$ that a message will be inserted to *Set* from the application instead. So, on average, only $1 - \frac{1}{\rho}$ of the peers with an empty *Set*, create a dummy message. This gives us the next equation: $\frac{N(\rho - H)}{\rho H} = s_0 N(1 - \frac{1}{\rho})$. And so, we get that $s_0 = \frac{\rho - H}{H(\rho - 1)}$. ■

Lemma 7: $S = \sum_{i=1}^{\infty} N_i \cdot i = N \cdot \sum_{i=1}^{\infty} s_i \cdot i$.

Proof: Directly from the definitions of N_i and s_i. ■

Lemma 8: $IC = \frac{SH}{N} + 1$.

Proof: By Little law, the number of messages in the network, S, equals to the incoming messages rate, $\frac{N}{H}$, times the average number of iterations a message spends in the network. Because we also count the last iteration (when the message leaves the system), we get the next equation: $S = \frac{N}{H}(IC - 1)$. ■

F. Calculating the distribution of Sets' sizes in the network

There are two methods to find $\{s_i\}_{i=0}^{\infty}$: simulations or analysis (queuing theory). We present a simple algorithm to find the distribution of Sets' sizes, with excellent fit to the simulations (see Section IV-D2). The idea of the algorithm is to look only on one *Set* and analyze the probabilities of changes in its size, and it is based on two simplifying.

1) While analyzing the number of messages that might arrive to *Set* we take the mean of the messages that were sent in some iteration and are not outgoing from the network in that iteration (messages that will not reach their destination in that iteration). Of course we consider only messages that are sent from the other $(N-1)$ *Set*s. We notate the mean of messages that might arrive to a peer in an iteration by D, and from Lemma 4, we get that $D = (N-1)(1 - \frac{1}{H})$.
2) We calculate s_i only for $i < n$, for a parameter n.

The algorithm builds a matrix M in the following way: $M_{i,j}$ is the probability that *Set* of size j will be of size i in the next iteration. Ideally, M is an infinite matrix, for our algorithm we choose its size, n. The accuracy increases as n increases.

The function **ProbToGetK**(int k) (Alg 3) returns the probability for a peer to get k messages in an iteration.

Algorithm 3 The **ProbToGetK**(int k) function.

return $\binom{D}{k}\left(\frac{1}{N-1}\right)^k\left(1 - \frac{1}{N-1}\right)^{D-k}$

Because D is a rational number, we should calculate the factorials in the binomial coefficient with Gamma function.

The function **ProbToBeChangedInK**(int k) (Alg 4) returns the probability for non zero *Set* size to be changed in k between two consecutive iterations. The minimal value for k that might return nonzero value is -1 (when both no message was sent to the peer and no message was inserted from the application); the corresponding maximal value, should be $N-1$, but due to our lenient assumption, we return 0 for every value that is greater than D.

Due to the range of the possible changes in *Set*'s size, M is almost lower triangular matrix; above the diagonal, only the values on the upper secondary diagonal are not zeros.

Algorithm 4 The **ProbToBeChangedInK**(int k) function.

if $k < -1$ OR $k > D$ **then**
  return 0.
else if $k = -1$ **then**
  return $(1 - \frac{1}{\rho})$**ProbToGetK**(0).
else
  return $\frac{1}{\rho}$**ProbToGetK**(k)+$(1 - \frac{1}{\rho})$**ProbToGetK**$(k+1)$.
end if

Therefore, for calculating M, we should run on the non-zero cells of each column (starting from the first) and fill them as in the **BuildProbsMatrix** function (Alg 5).

Algorithm 5 The **BuildProbsMatrix**(int n) function.

Initialize zeros matrix $M^{n \times n}$
for $i = 0$ to $n - 1$ **do**
  $M_{i,0}$ =**ProbToGetK**(i)
  $M_{i,1}$ =**ProbToBeChangedInK**$(i - 1)$
end for
for $j = 2$ to $n - 1$ **do**
  for $i = j - 1$ to $n - 1$ **do**
    $M_{i,j} = M_{i-1,j-1}$
  end for
end for
return M

For $M_{i,j}$, $i \geq 1, j \geq 2$, **BuildProbsMatrix** avoids unnecessary calculations: while $j > 2$, $M_{i,j} = M_{i-1,j-1}$ because $i - j = (i - 1) - (j - 1)$.

Now we define the vector $\overrightarrow{s} = (s_0, s_1, ..., s_{n-1})$. M that is returned from the **BuildProbsMatrix** function, defines the connections between the s_i values. For getting $\overrightarrow{s}$, we need to solve the next linear equations system: $M\overrightarrow{s}=\overrightarrow{s}$. This is a system of n equation and n variables. But, by Lemma 6, we know how to calculate s_0, so we can efficiently solve this system by forward substitution as could be seen in **DURPSetsSizesDist**(n) (Figure 6).

The time and space complexity of the algorithm are $\mathcal{O}(n^2)$, but the space complexity can be improved to $\mathcal{O}(n)$. In the end of the next section, we compare simulation results to the algorithm.

Algorithm 6 DURPSetsSizesDist(n). Efficient algorithm for getting the *Set*s' sizes distribution under DURP$^{\Pi}$. n is the accuracy parameter.

$M^{n \times n}$ =**BuildProbsMatrix**(n)
Initialize vector $\overrightarrow{s}^{n \times 1}$
$s_0 = \frac{\rho - H}{H(\rho - 1)}$
for $i = 1$ to $n - 1$ **do**
  $s_i = \frac{1}{M_{i-1,i}}(s_{i-1} - \sum_{j=0}^{i-1} M_{i-1,j} s_j)$
end for
return $\overrightarrow{s}$

IV. DURP's Performance

We now analyze DURP's efficiency: communication overhead and latency. We also compare these properties with the DC-net protocol [2]. In the end of this section, we bring simulation results for DURP^{OR} and DURP^{Crowds}.

A. DURP's Communication Overhead

We compare the communication overhead with other protocols that use constant sending rate to ensure unobservability. Therefore, we test the communication by the maximal application messages that might reach the destination, per sending actions. Every DURP's iteration, N sending actions occur. From Lemma 4, and from the proportion between application messages and dummy ones, (Lemma 5), we get that by mean, every iteration $\frac{N}{\rho}$ application messages reach the destination. Therefore, For sending a message anonymously, instead of one sending action, we pay in ρ actions. Intuitively, we expected this result, because we took a message from the application with probability of $\frac{1}{\rho}$.

The ratio of ρ sending actions per real message is the minimal, and is gotten if the application always has something to send. If peers do not have something to send, the actual ratio increases.

In the DC-net protocol [2] the minimal ratio is $\mathcal{O}(n)$.

B. DURP's Latency

We measure the latency of DURP by IC, the average *iterations-counter*. Although we have a formula for IC (Lemma 8), we cannot simply use it, as the calculation of S requires a run of DURPSetsSizesDist (Alg 6). However, the gotten IC value is a constant, and in the simulation results in the end of the section, we show configurations where IC is less than $10 \cdot H$, when H is the expected value for IC in the base protocol (in base protocols like OR and Crowds, a message is sent every iteration until it reaches its destination).

In the DC-net protocol, if we ignore the overhead of collusions, the latency is minimal.

C. Overhead vs. Latency, and DURP's Parameters

We showed that the communication overhead is determined by the ρ parameter. However, we can increase ρ and pay in more dummy messages (Lemma 5), and decrease the latency (IC). Another way to decrease the latency, is decreasing of H (for example, using lower P_f in Crowds, or fewer encryption layers in OR). However, this might hurt the anonymity against coalition of malicious destination and peers.

In the DC-net protocol, we pay for the low latency in significant communication overhead. Herbivore [11], a DC-net based protocol, pays for the scalability and the improvement in the communication overhead, in latency and in the loss of the unobservability against strong attackers.

D. DURP^{OR} and DURP^{Crowds} Simulation Results

1) Onion Routing and Crowds: Due to space limitations, we refer the reader to [4] and [3] for details about these protocols. We only shortly mention the routing procedure of both the protocol, as it is relevant for the simulation results. In OR, l intermediate peers are chosen to be the middle of the path between the sender and the destination. Because the *hops-counter* for all the messages is the same: $H = l+1$. In Crowds,

TABLE I
$N = 100, H = 4, \rho = 5$

	Expected results	DURP^{OR}	DURP^{Crowds}
$Real/Dummy$	4	4.00±0.00	4.00±0.00
N_0	6.25	6.25±0.00	6.25±0.00
S	929.92	934.05±0.57	933.74±0.67
IC	38.20	38.36±0.02	38.35±0.03
$MaxIC$	-	1128±62.68	1256.25±116.07

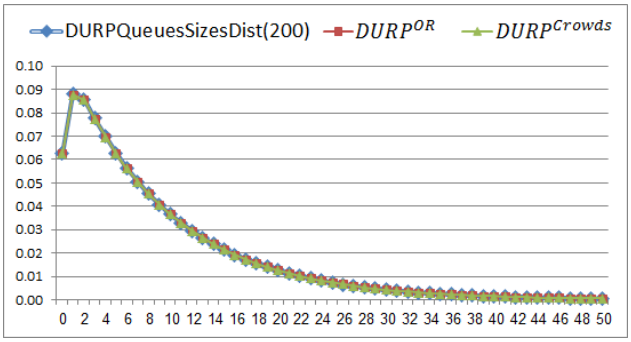

Fig. 1. Comparison of the *Sets*' sizes distribution. $N = 100, H = 4, \rho = 5$.

TABLE II
$N = 100, H = 5, \rho = 7$

	Expected results	DURP^{OR}	DURP^{Crowds}
$Real/Dummy$	2.5	2.50±0.00	2.50±0.00
N_0	6.67	6.67±0.00	6.67±0.00
S	832.93	835.71±0.47	835.51±0.41
IC	42.65	42.79±0.02	42.78±0.02
$MaxIC$	-	1062±137.87	1078.88±89.18

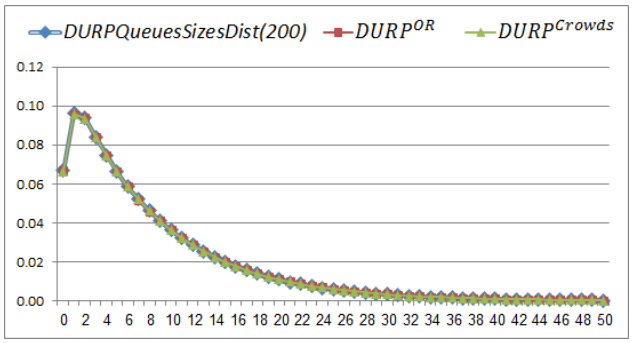

Fig. 2. Comparison of the *Sets*' sizes distribution. $N = 100, H = 5, \rho = 7$.

the route is chosen dynamically: The first intermediate peer is chosen by the sender, and from that point, the message is forwarded to another peer with probability $\frac{1}{2} < P_f < 1$, or sent to the destination with probability $1 - P_f$. Therefore, $H = 1 + \frac{1}{1-P_f}$.

Both the protocols ensure anonymity against the destination, but do not achieve unobservability, even against local eavesdropper attacker, as they do not use cover traffic.

2) Simulation results: We ran simulations of DURP^{Crowds} and DURP^{OR}, with different H, ρ and N values. The very high correlation between the actual results and the analysis of the previous section, shows that the algorithm for getting the distribution of the $Sets$' sizes (Alg 6) gives excellent results, and that using the algorithm and the lemmas, it is possible to predict network properties like IC, given DURP's configuration parameters (ρ, H).

We present the results of simulations for two different configurations. For every configuration, we compare the simulation results of both the base protocols that have identical expected results while they have the same *hops-counter* mean, H, to the expected results. We ran every simulation 8 times for 10,000,000 iterations, and after 1,000,000 iterations that started from network of peers with empty *Sets*.

The expected results are calculated as follows: The proportion between the application messages (reals) to the protocol dummies is calculated by Lemma 5. $N_0 = N \cdot s_0$ is calculated by Lemma 6. To calculate S by Lemma 7, we used **DURPSetsSizesDist** (Alg 6) with $n = 200$ to get s_i for $i = 0, ..., 199$. In the end, we calculate IC by Lemma 8.

$MaxIC$ is the maximal value of *iterations-counter* for message in the simulation. In the comparison tables, the confidence interval is calculated by confidence level of 99.9%.

1) $N = 100, H = 4, \rho = 5$. The comparison appears in Table I and in Figure 1.
2) $N = 100, H = 5, \rho = 7$. The comparison appears in Table II and in Figure 2.

In the Crowds simulations, the confidence interval of the average *hops-counter* was less than 0.001 around the expected value. The average maximal *hops-counter* was 53.25 in the first case, and 72 in the second.

V. The Effect of DURP$^{\Pi}$'s Networking Properties on DURP$^{\Pi}$'s Anonymity and Security

In this section we discuss possible problems and attacks against DURP$^{\Pi}$ that are effected by DURP$^{\Pi}$'s networking properties.

A. Degradation of Service Attack

In DURP$^{\Pi}$, the destination server has an expected value for the number of incoming messages every iteration. Flooding the network does not necessarily affect the rate of the incoming messages to the destination because all the peers send one message per iteration, regardless their *Set*'s size. Additionally, due to the distribution of the *Sets*' sizes and the high variance (see Figures 1, 2), it is hard for flooded peers to detect anomalies. We would like to have indications for such attacks from both the destination and the peers perspectives.

B. The Sets' Sizes

We separate this problem to two subproblems:

1) Too many empty Sets: The probability to have an empty *Set* is not low enough. There could be some implications for this fact. First, if an attacker blocks the incoming traffic to peer for low number of iterations, the peer cannot detect that (because it's normal to have an empty *Set*). Second, it is easy for a local eavesdropper to know when a peer is the source of messages it sends (even if the attacker does not know whether it is a real message or dummy).

2) The Sets' sizes are too low: As could be seen from the *Sets*' sizes distribution graphs (Fig 1 and 2), most of the *Sets* are of low size.

3) Example: We demonstrate the affect of low average *Set* size and low number of empty *Set*s, by analysis of DURP^{Crowds} against destination with local eavesdropping ability in the following example:

In the example, we show how the *Set*s' sizes distribution and in particular the number of empty *Set*s might affect the anonymity degree of DURP$^{\Pi}$. In this example we examine DURP^{Crowds} in network of N peers, and known destination which is not a peer in the network. The attacker is the single destination that has also the ability to eavesdrop the incoming and outgoing traffic of a peer I (Initiator). The original Crowds protocol, does not offer any protection against local eavesdropper, particularly when the destination is known. We prove that DURP^{Crowds} offers some protection against such attacker. We consider the next extreme situation: in the last X iterations, there was not any traffic to I (it can also happen due to blocking of the incoming traffic). So, the attacker knows that any outgoing message from I is created by I, with very high probability (in the analysis we take this as a fact). The attacker sees that I sends his message, marked by m, to another peer P.

We notate m_i as the message that P sends in the i-th iteration. We start counting from the iteration when I sent m to P (this is the iteration with index 0). The destination server, D, gets from P a message, m_j at the j-th iteration ($j > 0$).

Due to the fact that the communication between I and P is encrypted with their common private key, the attacker cannot learn from the packets content any information about whether m_j is m. We want to calculate the probability that given the fact that m_j was sent to D, $m_j = m$.

In this analysis we ignore the low probability that m was sent from P to another peer and was sent back to P until the j-th iteration (it does not change the conclusion).

We notate the average queue size with s, and the average fraction of messages with *hops-counter* i, by H_i (we need H_0). We want to calculate:

$$Pr\left[m_j = m | P \overset{m_j}{\rightarrow} D\right] \tag{1}$$

Using Bayes' law:

$$(1) = \frac{Pr\left[P \overset{m_j}{\rightarrow} D | m_j = m\right] \cdot Pr[m_j = m]}{Pr[P \overset{m_j}{\rightarrow} D]} \tag{2}$$

Given the fact that m is the message that P will send, the probability that it will be sent to D:

$$Pr\left[P \overset{m_j}{\rightarrow} D | m_j = m\right] = 1 - P_f \tag{3}$$

The probability that on the j-th iteration, P sends m:

$$Pr[m_j = m] = \left(\frac{s-1}{s}\right)^{j-1} \cdot \frac{1}{s} \tag{4}$$

For sending a message to D in some iteration, P should choose a message that he got (a message that was already sent), and that this message will not be forwarded:

$$Pr[P \overset{m_j}{\rightarrow} D] = (1 - H_0) \cdot (1 - P_f) \tag{5}$$

Substitution of Equations (3), (4) and (5) in (2) gives:

$$Pr\left[m_j = m | P \overset{m_j}{\rightarrow} D\right] = \frac{\left(\frac{s-1}{s}\right)^{j-1} \cdot \frac{1}{s}}{(1 - H_0)}$$

We can see that the probability of the attacker to identify I decreases when s increases. Additionally, without the possible assumption about the empty *Set* of I, we could decrease the above expression by multiplying it with the probability that m was originated by I.

C. The Maximal Iterations-Counter Value is Too High

As could be seen in the Tables I and II, the maximal *iterations-counter* is high. We can fix this by changing the choosing algorithm. From simulations we ran, FIFO, or uniform choosing from the first X messages in *Set* ($X \in \{3, 4, 5\}$) improved dramatically the maximal *iterations-counter* value. Yet, we do not discuss these results in this paper, because it is obvious that we prefer to choose uniformly from *Set*, for maximizing the mix affect.

VI. Optimizing the Design

In this section we offer some advanced versions of the basic DURP. We first offer a little change, and prove shortly that it solves one of the problems of the previous section, then we offer an optimization that solves all that problems.

A. Decrease the Empty Sets

For decreasing the mean of the number of empty *Set*s in the network, we just have to ask every peer, to do the round procedure (see Alg 1) more than one time every iteration. We denote DURP$^{\Pi}$ that does the round procedure i times by DURP$^{\Pi,i}$, and the fraction of *Set*s of size j in DURP$^{\Pi,i}$, with s^i_j. The next theorem, says that this change decreases the number of empty *Set*s:

Theorem 9: For every $\rho > 1$, and for every $i > 1 \in N$, $s^1_0 < s^i_0$.

Proof: From lemma (6) (where $i = 1$):

$$s^1_0 = \frac{\rho - H}{H(\rho - 1)} \tag{6}$$

Regardless to the probabilities of $Queue$ of size x to be changed in some y, and similarly to the proof of lemma (6) we can express s^i_0. We denote the mean of dummy messages that are added to the network every iteration with d. The mean of messages from the application that are added every iteration is $i\frac{N}{\rho}$. The proportion between the real messages to the dummy ones, is still $\frac{H}{\rho - H}$. So we get:

$$d = \frac{iN}{\rho} \cdot \frac{\rho - H}{H} \tag{7}$$

Another option to calculate d, is to sum the mean of dummy messages that every $Queue$ add every iteration. If there are i actions per iteration, then all the peers with $Queue$ of size $< i$ might add a dummy message. Every action, in probability of $\frac{1}{\rho}$ one message is added from the application, and one message is sent (the size of $Queue$ is not changed). In probability of $1 - \frac{1}{\rho}$ no message is added from the application, and one message is sent (the size of $Queue$ decreases by one). We

now check how many messages will be added if in exactly $j \leq i$ of the i actions, no application message was added. The probability for this to happen is $\binom{i}{j}(1-\frac{1}{\rho})^j(\frac{1}{\rho})^{i-j}$. In such situation, every peer with $Queue$ of size $k < j$ has to create and send $j-k$. The fraction of peers with $Queue$ of size k by mean, is defined as s_k^i. Generally, For k and j, such that $k < j \leq i$, if in j of the i $iteration-actions$ no message was added from the application queue, $Queue$ of size k will send $j-k$ dummy messages. Therefore:

$$d = N \cdot \sum_{j=1}^{i} \binom{i}{j}(1-\frac{1}{\rho})^j(\frac{1}{\rho})^{i-j}(\sum_{k=0}^{j-1}(j-k)s_k^i) \quad (8)$$

We multiply the right side in N, because we want to get the mean of dummy messages that are added every iteration in all the system.

We can rewrite (8) by swapping the $\sum$s on the right side:

$$\begin{aligned}(8) &= N \cdot \sum_{k=0}^{i-1} s_k^i(\sum_{j=k+1}^{i}(j-k)\binom{i}{j}(1-\frac{1}{\rho})^j(\frac{1}{\rho})^{i-j}) \\ &= N \cdot \sum_{k=0}^{i-1} s_k^i(\sum_{j=k+1}^{i}(j-k)\binom{i}{j}\frac{(\rho-1)^j}{\rho^i}) \\ &= \frac{N}{\rho^i} \cdot \sum_{k=0}^{i-1} s_k^i(\sum_{j=k+1}^{i}(j-k)\binom{i}{j}(\rho-1)^j) \quad (9)\end{aligned}$$

From comparing (7) and (9) we get:

$$\begin{aligned}\frac{i(\rho-H)}{H} &= \frac{1}{\rho^{i-1}} \cdot \sum_{k=0}^{i-1} s_k^i(\sum_{j=k+1}^{i}(j-k)\binom{i}{j}(\rho-1)^j) \\ &= s_0^i \frac{1}{\rho^{i-1}}\sum_{j=1}^{i} j\binom{i}{j}(\rho-1)^j + \\ &\frac{1}{\rho^{i-1}} \cdot \sum_{k=1}^{i-1} s_k^i(\sum_{j=k+1}^{i}(j-k)\binom{i}{j}(\rho-1)^j) \quad (10)\end{aligned}$$

Isolate s_0^i in (10):

$$s_0^i = \frac{i(\rho-H)}{H} \cdot \frac{\rho^{i-1}}{\sum_{j=1}^{i} j\binom{i}{j}(\rho-1)^j} - X \quad (11)$$

When X is the second element in the right side of (10), divided by the coefficient of s_0^i in (10). Obviously, $X > 0$.

By taking out of $\sum (\rho-1)$, and using (6):

$$\begin{aligned}s_0^i &= \frac{\rho-H}{H(\rho-1)} \cdot \frac{i \cdot \rho^{i-1}}{\sum_{j=1}^{i} j\binom{i}{j}(\rho-1)^{j-1}} - X \\ s_0^i &= s_0^1 \cdot \frac{i \cdot \rho^{i-1}}{\sum_{j=1}^{i} j\binom{i}{j}(\rho-1)^{j-1}} - X \quad (12)\end{aligned}$$

The next lemma, is for proving that the coefficient of s_0^1 in (12) is 1:

Lemma 10: $\forall \rho, i > 1 : i\rho^{i-1} = \sum_{j=1}^{i} j\binom{i}{j}(\rho-1)^{j-1}$.

Proof: By the binomial formula:

$$\begin{aligned}i\rho^{i-1} &= i\sum_{j=0}^{i-1}\binom{i-1}{j}(\rho-1)^j = i\sum_{j=1}^{i}\binom{i-1}{j-1}(\rho-1)^{j-1} \\ &= \sum_{j=1}^{i} i\frac{j}{j}\binom{i-1}{j-1}(\rho-1)^{j-1} = \sum_{j=1}^{i} j\binom{i}{j}(\rho-1)^{j-1}\end{aligned}$$

■

From the above lemma and (12), and due to the fact that $X > 0$, the theorem is proved. ■

B. The On Load Send More (OLSM) Optimization

This optimization gets three parameters:

1) $MinSize$. Minimal size for a peer to run the optimization.
2) T. The number of tries to send another message.
3) P. The probability that single try succeeds.

The optimization works as follows: if after running the round procedure (see Alg 1), a peer still has *Set* with size $\geq MinSize$, then he gets T chances to send another messages. Every chance, a random number, $0 \leq r \leq 1$, is chosen, and if $r \leq P$, then another message is sent from *Set* (see Alg 7). We denote DURP$^{\Pi}$ with the OLSM optimization by DURP$^{\Pi}_{OLSM}$.

Algorithm 7 The OLSM optimization is called right after the round procedure (Alg 1).

```
if Set.size ≥ MinSize then
  for i = 1 to T do
    r ←r [0, 1]
    if r ≤ P then
      (c, p) ←r Set
      Send c to p
    end if
  end for
end if
```

For using DURP$^{\Pi}_{OLSM}$ and getting its properties, some configurations of DURP$^{\Pi}$ should be changed. In DURP$^{\Pi}$ we demanded that $\rho - H > 0$ for the network stability (Lemma 3). DURP$^{\Pi}_{OLSM}$ demands the opposite: $H - \rho > 0$. The meaning of this demand is that without the optimization, S does not converge ($S = \infty$). For ensuring the convergence of S with the optimization, we also demand: $\frac{1+T \cdot P}{H} - \frac{1}{\rho} > 0$. Shortly, we demand that if all the *Set*s' sizes are greater than $MinSize$, than the mean of outgoing messages (by the round procedure and by the OLSM optimization) is greater than the mean of the new messages. We also demand $MizSize \geq T$.

1) DURP$^{\Pi}_{OLSM}$ properties: We now present some lemmas without their proofs; some of the proofs are similar to proofs of similar properties in DURP$^{\Pi}$, and some proofs are trivial. Anyway, we present them such that no lemma depends on its follows:

Lemma 11: The mean of the sending actions during one OLSM call is $T \cdot P$.

Lemma 12: A network under DURP$^{\Pi}_{OLSM}$ is *stable* when $\frac{1+T \cdot P}{H} - \frac{1}{\rho} > 0$.

All the below lemmas, assume Lemma 12's condition.

Lemma 13: The average number of the outgoing messages (messages that arrive to the destination) and the average number of new messages every iteration are $\frac{N}{\rho}$.

Lemma 14: The average *iterations-counter* $IC = \frac{\rho \cdot S}{N} + 1$.

Lemma 15: The number of 'send' events in the round procedure every iteration is N (exactly as in DURP$^{\Pi}$).

Lemma 16: The average number of 'send' events due to the OLSM optimization every iteration is $H\frac{N}{\rho} - N$. We denote this value as $OLSM_{hops}$.

Lemma 17: The average number of OLSM calls every iteration, denoted by $OLSM_{calls}$, is $\frac{1}{T \cdot P} OLSM_{hops}$.

Lemma 18: $\frac{1}{\rho} s_{MinSize} + \sum_{i=MinSize+1}^{\infty} s_i = \frac{OLSM_{calls}}{N}$.

The last lemma stems from the fact that the OLSM optimization is called only if the *Set*'s size is greater than $MinSize$ or if *Set*'s size equals $MinSize$ and a message was added to *Set* from the application (probability of $\frac{1}{\rho}$).

2) Algorithm for getting the distribution of the sizes of the Sets: In Section III-F we presented an algorithm for getting distribution of the sizes of the *Set*s. We cannot use this algorithm when the OLSM optimization is on. We now present similar algorithm, that works also for DURP$^{\Pi}$ (just set the $MinSize$ parameter to ∞). As in DURP$^{\Pi}$, we notate the mean of messages that might arrive to some peer (not the destination) as D. D is calculated by substraction of the outgoing messages' mean from the mean of the total sending actions in iteration when we count only all the other peers messages (we consider only messages that might arrive to the peer). $D = \frac{N-1}{\rho} H - \frac{N-1}{\rho} = \frac{(N-1)(H-1)}{\rho}$.

The function **OlsmProbToSendK**(int k) (Alg 8) returns the probability that during a call to the OLSM optimization, k messages were sent from the caller's *Set*.

Algorithm 8 The **OlsmProbToSendK**(int k) function.

return $\binom{T}{k} P^k (1-P)^{T-k}$

Let's denote the difference between the incoming messages (calculated by **ProbToGetK**, Alg 3) and the outgoing messages due to OLSM call (calculated by **OlsmProbToSendK**, Alg 8) by Δ. The function **OlsmΔProb**(int k) (Alg 9) returns the probability that during a call to the OLSM optimization $\Delta = k$.

Algorithm 9 The **OlsmΔProb**(int k) function.

output $= 0$
for $i = 0$ to T **do**
  output += **OlsmProbToSendK**(i)·**ProbToGetK**($i + k$)
end for
return output

Similarly to DURP$^{\Pi}$, we build the probabilities transitions matrix, M, using the **OlsmBuildProbsMatrix** function (Alg 10). As in DURP$^{\Pi}$, with a bit longer code, we could avoid recalculation of the same values.

Now we define the vector $\overrightarrow{s} = (s_0, s_1, ..., s_{n-1})$. M that is returned from the **OlsmBuildProbsMatrix** function, defines the connections between all the s_i values. Exactly as in DURP$^{\Pi}$, we solve the linear equations system: $M\overrightarrow{s} = \overrightarrow{s}$. We want to find the eigenvector for the eigenvalue $\lambda = 1$. If we look at the infinite transitions probabilities matrix, it's easy to see that it has eigenvalue $\lambda = 1$, because the sum of every column is 1. We use finite square matrix of size n, and if n is high enough, this matrix also has self value 1 or self value λ such that $|1-\lambda|$ is very small. So, for getting the sizes distribution, the **OLSMSetsSizesDist** (Alg 11) algorithm finds the the eigenvector that corresponds to λ, and normalizes it such that the sum of its elements will be 1. The normalized vector is the wanted distribution.

Algorithm 10 The **OlsmBuildProbsMatrix**(int n) function.

Initialize zeros matrix $M^{n \times n}$
for j=0 to n-1 **do**
  for i=j-1 to n-1 **do**
    if $j = 0$ **then**
      $M_{i,j}$=**ProbToGetK**(i)
    else if $j < MinSize$ **then**
      $M_{i,j}$=**ProbToBeChangedInK**(i-j)
    else if $j = MinSize$ **then**
      $M_{i,j}$=$\frac{1}{\rho}$**OlsmΔProb**(i-j)+(1-$\frac{1}{\rho}$)**ProbToGetK**(i-j+1)
    else if $j > MinSize$ **then**
      $M_{i,j}$=$\frac{1}{\rho}$**OlsmΔProb**(i-j)+(1-$\frac{1}{\rho}$)**OlsmΔProb**(i-j+1)
    end if
  end for
end for
return M

Algorithm 11 OLSMSetsSizesDist(int n). Algorithm for getting the *Set*s' sizes distribution under DURP$^{\Pi}_{OLSM}$. n is the accuracy parameter.

$M^{n \times n}$ =**OlsmBuildProbsMatrix**(n)
$\lambda \leftarrow$ The eigenvalue, l, that minimize $|l - 1|$.
$\overrightarrow{s} \leftarrow$ Eigenvector of λ
$sum \leftarrow$ Sum of $\overrightarrow{s}$ elements.
for $i = 0$ to $n - 1$ **do**
  $s_i = s_i / sum$
end for
return $\overrightarrow{s}$

C. OLSM Simulation Results

We ran the simulations for DURP$^{\Pi}_{OLSM}$ as we did for DURP$^{\Pi}$ (Section IV-D2). The expected results are calculated as follows: The **OLSMSetsSizesDist** algorithm (Alg 11) with $n = 100$ outputs s_i for $i = 0, ..., 99$. S is calculated by Lemma 7, and IC by Lemma 14. $MaxIC$ is the maximal *iterations-counter* value for a message in the simulation. $OLSM_{hops}$ and $OLSM_{calls}$ are calculated by Lemmas 17 and 18.

1) *N=100, H=4, ρ=3, MinSize=20, T=4, P=0.5.* The comparison appears in Table III and in Figure 3.
2) *N=100, H=4, ρ=2, MinSize=15, T=5, P=0.5.* The comparison appears in Table IV and in Figure 4.

In the Crowds simulations, the confidence interval of the average *hops-counter* (H) was less than 0.001 around the expected value. The average maximal *hops-counter* was 50.5 in the first case, and 52.87 in the second.

D. DURP$^{\Pi}_{OLSM}$ Advantages

In Section V we presented three problems of DURP$^{\Pi}$; DURP$^{\Pi}_{OLSM}$ provides solutions to these problems:

TABLE III
$N = 100, H = 4, \rho = 3, MinSize = 20, T = 4, P = 0.5$

	Expected results	DURP^{OR}_{OLSM}	$\text{DURP}^{Crowds}_{OLSM}$
S	1833.34	1832.85±0.06	1832.75±0.05
IC	56.00	55.99±0.00	55.98±0.00
$MaxIC$	-	404.5±35.83	864.75±70.27
$OLSM_{hops}$	33.33	33.33±0.01	33.33±0.01
$OLSM_{calls}$	16.67	16.67±0.00	16.67±0.00

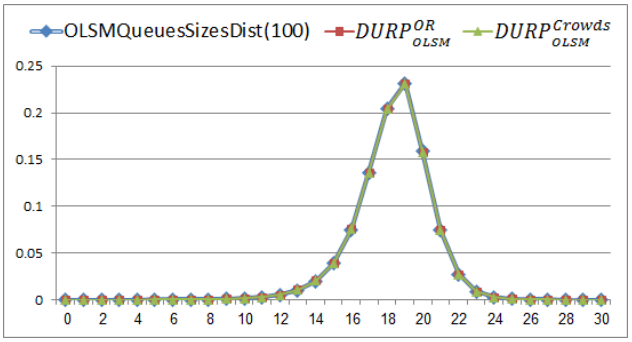

Fig. 3. Comparison of the *Sets*' sizes distribution. $N = 100, H = 4, \rho = 3, MinSize = 20, T = 4, P = 0.5$.

TABLE IV
$N = 100, H = 4, \rho = 2, MinSize = 15, T = 5, P = 0.5$

	Expected results	DURP^{OR}_{OLSM}	$\text{DURP}^{Crowds}_{OLSM}$
S	1463.42	1463.56±0.02	1463.56±0.02
IC	30.27	30.27±0.00	30.27±0.00
$MaxIC$	-	215.75±17.75	490.25±34.29
$OLSM_{hops}$	100	100.00±0.01	100.00±0.01
$OLSM_{calls}$	40	40.00±0.00	40.00±0.00

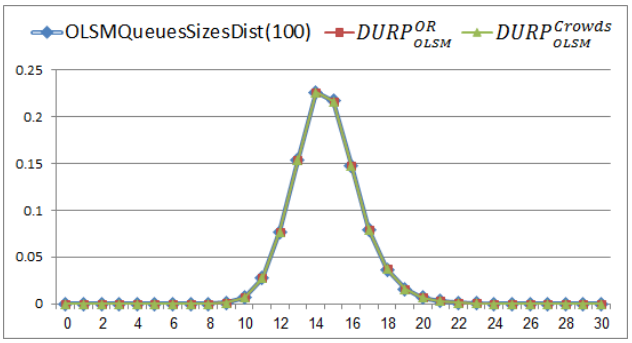

Fig. 4. Comparison of the *Sets*' sizes distribution. $N = 100, H = 4, \rho = 2, MinSize = 15, T = 5, P = 0.5$.

As could be seen in Tables III and IV, the maximal *iterations-counter* value was decreased.

Additionally, with the OLSM optimization, if the network is flooded, the destination gets more than $\frac{N}{\rho}$ messages in average, and from the peers perspective, because the variance of the *Sets*' sizes distribution was decreased dramatically (See Figures 3 and 4), it is possible to detect anomalies by the *Set*'s size.

Figures 3 and 4 show that DURP^{Π}_{OLSM} can be configured such that both s_0 is almost 0, and the average *Set* size isn't low.

Another direct result of the negligible s_0 value, is negligible amount of dummy messages; this means that unlike DURP^{Π}, in DURP^{Π}_{OLSM}, the redundancy is negligible while the application has messages to send in rate of $\frac{1}{\rho}$. On the other hand, in DURP^{Π}_{OLSM}, there is more traffic than DURP^{Π}; for decreasing the communication volume, we can increase proportionally the interval between every two iterations.

The OLSM optimization, does not hurt the unobservability of DURP^{OR}_{OLSM} and $\text{DURP}^{Crowds}_{OLSM}$. The unobservability proofs for these protocols, are similar to the proof for DURP^{OR} and DURP^{Crowds} as brought in Appendix D.

VII. Conclusions and Future Research

We presented DURP, a modular protocol for anonymous communication, featuring low latency and overhead, together with source anonymity against malicious destination *and* strong unobservability (against global eavesdropper controlling many peers). We analyzed the network properties of DURP, with simplified analysis matching very well simulations, and of an optimized version, DURP_{OLSM}. Future research is required, to extend our work to model with imperfect synchronization, semi-asynchronous/asynchronous models, and to to allow unobservable, anonymous responses and connections.

Appendix A
Lemma 3's Proof

Proof: Assuming the network is not stable. Let's look at the whole network in terms of energy. Every new message in the system, has the same mean of energy, H. When a message is sent ($hopes - counter$ increases by one), the energy of the message, decreases by one. When a message reaches its destination and exit the network, it has no energy (this proof is enabled due to the restrictions on the base protocols in subsection III-D). Every iteration, the network decreases from the total energy N units, as the *hops-counter* of N messages increases by one. The mean of the incoming energy, is the total energy's mean of the new messages (from the application and dummies). Therefore:

$$H(\frac{N}{\rho} + N_0(1 - \frac{1}{\rho})) > N$$
$$s_0 = \frac{N_0}{N} > \frac{\rho - H}{H(1 - \rho)} \quad (13)$$

For calculating s_i, we should build the probabilities transitions graph (that could be represented also by infinite matrix). We showed how to build this graph in Alg 5, and the only change we should do, is to remove the lenient assumption we had, and that helped us to calculate the probability of $Queue$ to get k messages from the other $Queue$s between every two iterations (Alg 3). Let's denote the real function as **PTG**(k). Let G be the probabilities transitions graph; in G, every node represents s_i. for every $i > 0$, s_i has one backward to s_{i-1}, one edge to itself, and $N-1$ edges forward s_j, $i+1 \leq j \leq i+N-1$. s_0 has edge to itself, and $N+1$ edges forward. The sum of the probabilities on all the outgoing edges from every s_i is 1. Let's denote the mean of the fraction of the outgoing messages every iteration from the total sent messages every iteration (N), by X. We proved in lemma 4, that when the network is stable, $X = \frac{1}{H}$. We also proved in lemma 6

that if $X = \frac{1}{H}$, then $s_0 = \frac{\rho - H}{H(1-\rho)}$, and it's true also in the second direction: if $s_0 = \frac{\rho - H}{H(1-\rho)}$, than $X = \frac{1}{H}$. Therefore, if the network is not stable, then $X \neq \frac{1}{H}$. Additionally, $X \not\geq \frac{1}{H}$. Otherwise, the mean of the final *hops-counter* is less than H. Therefore, $X < \frac{1}{H}$.

Now we want to compare G in case of stable network ($X = \frac{1}{H}$) to unstable network ($X < \frac{1}{H}$). In both the cases, the number of messages that are sent every iteration is N. In the stable case, the mean of the messages that might arrive to the $Queue$ is lower than the unstable case, because in the stable case, more messages leave the network to the destination. Therefore, even without knowing the exact **PTG** function, we know for sure that all the backward edges in G (The edges that are directed closer to s_0), have lower probability when the network is unstable than in the case of stable network, because the probability to get some message is higher, and the $Queue$'s size is decreased by one only when no message arrive to the $Queue$. On the other hand, in the unstable network, there is a higher probability to move farther from s_0 in G, than in the case of stable network.

Therefore, if we look at both the solutions of the linear equation system that represent the transition probabilities in both the cases, s_0 in the stable network is higher than in the unstable network, and therefore for s_0 in the unstable network :

$$s_0 < \frac{\rho - H}{H(1-\rho)} \tag{14}$$

‘

In contradiction to (13). ■

APPENDIX B
COMP-N-ANONYMITY DEFINITIONS

Following Hevia and Micciancio [12], we offer definition that is based on an experiment that simulates protocol run over some network.

We let the adversary choose between two scenarios. The “relation” between the scenarios is restricted by the anonymity notion **N** that is tested in the experiment and by the capability of the attacker. The adversary controls the scenarios, by controlling the application level of all the protocol participants: who sends what to whom in both scenarios. This is done by periodically choosing two matrices of messages, $M^{(0)}$ and $M^{(1)}$, one for each scenario. We define two experiments. The first simulates the protocols by the $M^{(0)}$ matrices, and the second by $M^{(1)}$ matrices. The adversary, that gets information about the protocol simulation by its capability (for example: global eavesdropper gets all the traffic), has to distinguish between the experiment, by guessing which world was simulated.

A. The network model

Because our experiment simulates adaptive and active attacker, we need an accurate communication and execution model. For simplicity, in our network model there is a link in the experiment between every two participants, and every message between two participants is sent over their common link. We also assume the fully synchronous model. Namely, our experiments run in synchronized iterations.

B. The network peers

We model every peer by two levels; the application level: who want to send what to whom, and the protocol level: the execution of the application level wish by the protocol. The application level of all the participants is controlled by the attacker. The attacker chooses which messages are added to the application queues of all the peers, by choosing messages matrices. The application queue is managed by the FIFO algorithm, such that when peer gets a multiset of messages to send, it just enqueues the message into the application queue. It's important to remark that among the honest peers (peers that are not under the attacker's control), only the peers that need to send a message by the protocol will access their application queue, and only valid messages by the model and by the protocol (for example, messages with invalid destination) will be handled.

In the protocol level, the honest peers act by the protocol and are simulated by the experiment. On the other hand, the attacker can do whatever he can and wants with the machines it controls.

The peers might have different roles in the protocol; for example, protocols that use central mixes [14] [15] or routers [16] for helping the other peers to send anonymously, have two types of peers: client and mix (or router). The roles of the participants are determined by the protocol.

C. Attacker model

The experiment we present, can test a protocol against active malicious peers, eavesdroppers to peers and combinations of them. In Appendix C we extend the definition and the experiment to deal also with the attacker as malicious destination. The information that the attacker gets, depends on the peers it controls in the network, and on the peers it eavesdrops to them.

The attacker controls the application level of all the peers. Namely, the attacker chooses, for every peer, which messages to send and to whom. We only limit the messages to be valid by the tested protocol and by the model, and enforce the attacker that in both the scenarios, the messages matrices will not contain different unprotected data. In spite of these limitations, in the protocol level of the attacker's controlled peers, the attacker can deviate from the protocol.

The power to control the application, might seem extreme, but like the logic beyond the experiments for cryptographic encryption schemes [13] [17], and as mentioned in [12], in the reality, the attacker might have an influence on the application level of its victims. We conservatively give the attacker the whole control, as we cannot predict the attacker's influence about the application in different real scenarios.

D. Notations

1) Cryptographic and mathematical notations: We use the following common cryptographic and mathematical notations: $\mathcal{PPT}$ is probabilistic polynomial time. for $n \in \mathbb{N}$, $[n] \stackrel{def}{=} \{1, 2, ..., n\}$. $\mathcal{P}(S)$ is the power set of the set S. For any two sets, S_1 and S_2, $S_1 \in S_2*$ if and only for every $s \in S_1$, $s \in S_2$.

2) The experiment parameters: The first two parameters of the experiment, π and n, represent the protocol. π is a $\mathcal{PPT}$ algorithm that represents the tested protocol, and n is the number of participants in the protocol simulation, $n < l(k)$ when $l(\cdot)$ is some polynomial, and k is the security parameter of the experiment. π's *setup* method receives n as a parameter, and outputs a sequence of n states that represent the initial state of the machines in the protocol (denoted by $\{STATE_i\}_{i=1}^{n}$). Usually, we will want that π's *setup* method will do actions like keys, roles and identities distributions. π's *simulate* method receives a state of π participant with its incoming traffic and new messages from the application level, and returns its next state and its outgoing traffic.

The last experiment's parameters, $\mathcal{A}$ and *Cap*, define the attacker. $\mathcal{A}$ is the attacker $\mathcal{PPT}$ algorithm. *Cap* $\in \mathcal{P}([n])^2$ defines the attacker capabilities; *Cap* = $(\overline{H}, EV)$, such that $\overline{H}$ is the set of the indexes of the machines that are controlled by $\mathcal{A}$, and EV is the indexes set of the eavesdropped machines. We denote Cap's elements by *Cap*$[\overline{H}]$ and *Cap*$[EV]$ respectively.

An attacker with capability *Cap* = $(\overline{H}, EV)$, controls the machines with index $\in$ *Cap*$[\overline{H}]$ and eavesdrops the traffic of the machines with index $\in$ *Cap*$[EV]$.

In the next section, we extend the capability to deal also with malicious destination, by adding a bit MD.

3) Technical notions: Hevia and Micciancio [12] elegantly defined the scenarios by messages matrices: $V = \{0,1\}^l$ is the messages space, and a collection of messages between n parties is represented by $n \times n$ matrix, $M = (m_{i,j})_{i,j\in[n]}$. $m_{i,j} \in V^*$ is the multiset of messages from the i-th party to the j-th party [1].

4) Security notions: Following [12], the unprotected data (the attacker assumed knowledge), is defined by the functions $f_\cup$, f_Σ and $f_\#$ that map matrices from $\mathcal{M}_{n\times n}(V^*)$ into V^*, $\mathbb{N}^n$ and $\mathbb{N}$ respectively:

$$f_\cup(M) \stackrel{def}{=} (\uplus_{j\in[n]} m_{i,j})_{i\in[n]}$$

$$f_\Sigma(M) \stackrel{def}{=} (\sum_{j\in[n]} |m_{i,j}|)_{i\in[n]}$$

$$f_\#(M) \stackrel{def}{=} \sum_{i,j\in[n]} |m_{i,j}|$$

Additionally, we define $f_\cup^T(M) \stackrel{def}{=} f_\cup(M^T)$ and similarly $f_\Sigma^T(M) \stackrel{def}{=} f_\Sigma(M^T)$. For a given function $f \in \{f_\cup, f_\cup^T, f_\Sigma, f_\Sigma^T, f_\#\}$, the relation R_f on $\mathcal{M}_{n\times n}(V^*)^2$ is defined by $(M^{(0)}, M^{(1)}) \in R_f$ if and only if $f(M^{(0)}) = f(M^{(1)})$.

5) The $R_{\mathbf{N}}^H$ relation: The $R_{\mathbf{N}}$ relations in Table V (and [12]) are applicable only for passive adversaries. If the attacker controls a peer in the protocol, it can just inspect the messages in the peer's application queue and check whether they are from $M^{(0)}$ or from $M^{(1)}$. It can do the same also with the messages that the controlled peer receives. Consequently, the $R_{\mathbf{N}}$ relations, cannot be used for active adversaries.

[1]We replaced [12]'s notation $\mathcal{P}(V)$ with V^*, because powerset does not contains multisets

N	Notion	Definition of $R_{\mathbf{N}}$		
SUL	Sender Unlinkabilitiy	R_{SUL}	$\stackrel{def}{=}$	$R_{f_\Sigma} \cap R_{f_\cup^T}$
RUL	Receiver Unlinkabilitiy	R_{RUL}	$\stackrel{def}{=}$	$R_{f_\cup} \cap R_{f_\Sigma^T}$
UL	Unlinkabilitiy	R_{UL}	$\stackrel{def}{=}$	$R_{f_\Sigma} \cap R_{f_\Sigma^T}$
SA	Sender Anonymity	R_{SA}	$\stackrel{def}{=}$	$R_{f_\cup^T}$
RA	Receiver Anonymity	R_{RA}	$\stackrel{def}{=}$	$R_{f_\cup}$
SA*	Strong Sender Anonymity	R_{SA*}	$\stackrel{def}{=}$	$R_{f_\Sigma^T}$
RA*	Strong Receiver Anonymity	R_{RA*}	$\stackrel{def}{=}$	R_{f_Σ}
SRA	Sender-Receiver Anonymity	R_{SRA}	$\stackrel{def}{=}$	$R_{f_\#}$
UO	Unobservability	R_{UO}	$\stackrel{def}{=}$	$\mathcal{M}_{n\times n}(V^*)^2$

TABLE V
HEVIA AND MICCIANCIO'S [12] TABLE FOR ANONYMITY VARIANTS DEFINES EACH VARIANT **N** AND ITS ASSOCIATED RELATION $R_{\mathbf{N}}$

We address this by defining new relations family, named $R_{\mathbf{N}}^H \subseteq \mathcal{M}_{n\times n}(V^*)^2$.

Definition 19: For a given $n \in \mathbb{N}$, consider a pair of matrices, $(M^{(0)}, M^{(1)}) \in \mathcal{M}_{n\times n}(V^*)^2$, $H \subseteq [n]$, and a relation $R_{\mathbf{N}} \subseteq \mathcal{M}_{n\times n}(V^*)^2$. We say that $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}^H$ if and only if

1) $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}$.
2) For every $i \in [n] - H$ and $j \in [n]$, $M_{i,j}^{(0)} = M_{i,j}^{(1)}$ and $M_{j,i}^{(0)} = M_{j,i}^{(1)}$.

We do not forbid messages between honest and corrupted peers, as the attacker can legitimately get information from the times when messages are sent or received, and allow the attacker to add such messages, but with the restriction that in both the scenarios the messages must be identical. $R_{\mathbf{N}}^H$ extends the demand of identical unprotected data in both the matrices, to active attackers. If the attacker controls some machines in addition to their application level, the messages that they send, and suppose to receive are known to the attacker for any anonymity notion **N**.

In the experiment, H represents the honest peers set. Due to the demand that messages that are not between honest peers, will be identical in both the matrices, the $R_{\mathbf{N}}^H$ actually says that the $R_{\mathbf{N}}$ relation applies on the sub-matrices that represent only the messages between honest peers.

Figure 5 depicts the relation. In Definition 23 we extend $R_{\mathbf{N}}^H$ to $R_{\mathbf{N}}^{H,\tau}$, to strengthen attackers that are tested as malicious destination.

6) Experiment additional notations: $STATE_i$ is the state of the i-th participant. The experiment saves and manages the states of the honest participants.

$STATE_{\mathcal{A}}$ is the state of the attacker $\mathcal{A}$. The experiment gets and saves the attacker state after every action of $\mathcal{A}$, and sends it as a parameter for every action $\mathcal{A}$ should do. The initial information for $\mathcal{A}$ is the initial state of the peers it controls.

We use $C_{i,j,t}$ to denote the set of the elements (possibly ciphertexts), that were sent from the i-th participant to the j-th participants (the participants that are represented by $STATE_i$ and $STATE_j$) during the t-th iteration.

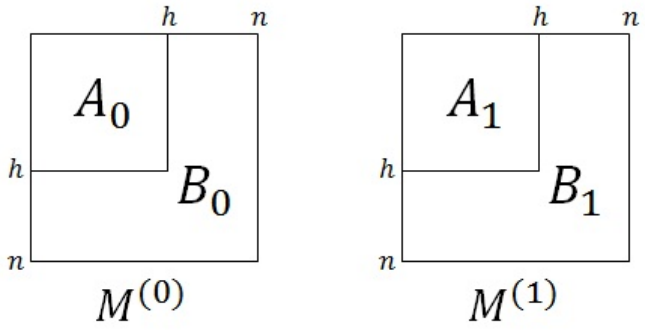

Fig. 5. Example of $R^H_{\mathbf{N}}$, for $H = [h] \subset [n]$. $(M^{(0)}, M^{(1)}) \in R^H_{\mathbf{N}}$ if and only if $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}$ and $B_0 = B_1$. Notice that $(A_0, A_1) \in R_{\mathbf{N}} \subseteq \mathcal{M}_{|H|\times|H|}(V^*)^2$.

E. The Experiment $Expt^{\mathbf{Comp-N-b}}_{\pi,n,\mathcal{A},Cap}(k)$

The experiment (see Alg 12) simulates the protocol, π, over one of two scenarios that the attacker, $\mathcal{A}$, chooses and manages adaptively. The simulated scenario is chosen by the b bit. At the beginning of the experiment, π's *setup* produces a sequence of initial states for all the simulation participants. The set of the participants' indexes that $\mathcal{A}$ gets their incoming and outgoing traffic is denote by EV, and the honest peers indexes set is denoted by H. Both the sets are determined by the Cap and n parameters. $\mathcal{A}$ is then initialized with the states of the participants it controls, and decides the maximal number of iterations that the experiment will run ($rounds \in poly(k)$, as $\mathcal{A}$ is a $\mathcal{PPT}$ algorithm, and it writes 1^{rounds}).

During the simulation, the attacker receives all the incoming and outgoing traffic of the controlled and eavesdropped participants. Every experiment's iteration, begins with an option for $\mathcal{A}$, to choose two messages matrices, $M^{(0)}$ and $M^{(1)}$. The experiment verifies that the matrices have identical unprotected data by the tested anonymity notion, **N** (verifies that $(M^{(0)}, M^{(1)}) \in R^H_{\mathbf{N}}$).

If the matrices are valid, the experiment passes only the messages in the $M^{(b)}$ matrix to the application queues of the participants and simulates the honest participants by π. $\mathcal{A}$ simulates the participants it controls (unnecessarily by the protocol).

At the end of every iteration, $\mathcal{A}$ has an opportunity to guess which scenario was simulated. If the attacker does not want to guess, it just returns $NULL$ into b'. When $\mathcal{A}$ chooses the bit b', the experiment is ended with the output b'. The experiment might be ended in returning 0 if the attacker chooses invalid pair of matrices ($\notin R^H_{\mathbf{N}}$), or after $rounds$ iterations.

Definition 20: The **Comp-N-advantage** of an attacker $\mathcal{A}$ that runs with k as a parameter, is defined as:

$$Adv^{Comp-\mathbf{N}}_{\pi,n,\mathcal{A},Cap}(k) =$$

$$|Pr[Expt^{Comp-\mathbf{N}-1}_{\pi,n,\mathcal{A},Cap}(k) = 1] - Pr[Expt^{Comp-\mathbf{N}-0}_{\pi,n,\mathcal{A},Cap}(k) = 1]|$$

Definition 21: Protocol π is Comp-**N**-anonymous, when $\mathbf{N} \in \{SUL, RUL, UL, SA, RA, SA^*, RA^*, SRA, UO\}$, against attackers with capability $Cap \in P([n])^2$, if for all $\mathcal{PPT}$ algorithms, $\mathcal{A}$, there exists a negligible function $negl$ such that,

$$Adv^{Comp-\mathbf{N}}_{\pi,n,\mathcal{A},Cap}(k) \leq negl(k)$$

Algorithm 12 $Expt^{\mathbf{Comp-N-b}}_{\pi,n,\mathcal{A},Cap}(k)$

1: $\{STATE_i\}^n_{i=1} \leftarrow \pi.Setup(1^k, n)$
2: $EV = Cap[\overline{H}] \cup Cap[\text{EV}]$
3: $H = [n] - Cap[\overline{H}]$
4: $< \quad STATE_{\mathcal{A}}, 1^{rounds} \quad > \leftarrow \mathcal{A}.Initialize(1^k, \{STATE_i\}_{i\in Cap[\overline{H}]})$
5: **for** $t = 1$ to $rounds$ **do**
6: $< \quad STATE_{\mathcal{A}}, M^{(0)}, M^{(1)} \quad > \leftarrow \mathcal{A}.InsertMessages(1^k, STATE_{\mathcal{A}})$
7: **if** $(M^{(0)}, M^{(1)}) \notin R^H_{\mathbf{N}}$ **then**
8: **return** 0
9: **end if**
10: **for all** $i \in H$ **do**
11: $< \quad STATE_i, \{C_{i,j,t}\}^n_{j=1} \quad > \leftarrow \pi.Simulate(1^k, STATE_i, \{C_{j,i,t-1}\}^n_{j=1}, \{m^{(b)}_{i,j}\}^n_{j=1})$
12: **end for**
13: $< \quad STATE_{\mathcal{A}}, \{C_{i,j,t}\}_{\substack{i\in Cap[\overline{H}] \\ 1\leq j\leq n \\ \text{or} \\ j\in EV}} \quad > \leftarrow \mathcal{A}.Simulate(1^k, STATE_{\mathcal{A}}, \{C_{i,j,t-1}\}_{i\in EV})$
14: $< STATE_{\mathcal{A}}, b' > \leftarrow A.GuessB(1^k, STATE_{\mathcal{A}})$
15: **if** $b' \neq NULL$ **then**
16: **return** b'
17: **end if**
18: **end for**
19: **return** 0

F. Experiment Run Time is $\mathcal{O}(poly(k))$

The total runtime of the experiment (Alg 12) is critical for the proof of the anonymity notions. In section D, we present a proof that is based on polynomial time reduction from the cryptographic primitives. The reduction contains simulation of the above experiment, and therefore its runtime must be polynomial in the security parameter k.

All the actions during the simulation take $\mathcal{O}(poly(k))$, and all the loops run for $\mathcal{O}(poly(k))$ iterations: The algorithms π and $\mathcal{A}$ are polytime, and all the other actions in the experiment take constant time. The main loop in the experiment does no more iterations than the length of a parameter that $\mathcal{A}$ outputs in $poly(k)$ time (during the $Initialize$ method with 1^k as the first argument), such that the loop's iterations can be bounded by some polynomial in k. the inner loop has no more than n iterations, and n is $poly(k)$. The attacker's total runtime is also polynomial in k, as the attacker's total runtime is bounded by the experiment's total runtime.

APPENDIX C
ANONYMITY AGAINST MALICIOUS DESTINATION

The Comp-**N**-anonymity definition of the previous section covers the following attackers: eavesdroppers, malicious peers, and any combination of them that controls the application adaptively. However, due to the restrictions of the $R^H_{\mathbf{N}}$ relation, the definition cannot be used for testing anonymity properties when the attacker controls destinations. Such an attacker model is relevant for anonymous mail services and anonymous

web surfing. Namely, this is the main goal of peer to peer networks like Tor [16] and services like Anonymizer [2].

In this section we extend Definition 21 to deal also with malicious destination. This extension is relevant only for three Comp-**N**-anonymity notions: $\mathbf{N} \in \{SUL, SA, SA*\}$. The other anonymity notions are aimed to hide information that the destination has, and therefore they are irrelevant in such an attacker model.

The intuition for our definition is to give the attacker additional power, by relaxing $R^H_{\mathbf{N}}$'s restrictions. We classify the messages in the matrices $M^{(b)}$ ($b \in \{0,1\}$) into three classes; each class can be modeled as $M^{(b)}$'s sub-matrix:

Sub-matrix notation. For $S_1, S_2 \subseteq [n]$ and $M \in \mathcal{M}_{n\times n}(V*)$, $M[S_1; S_2] \in \mathcal{M}_{|S_1|\times|S_2|}(V*)$ is M's sub-matrix such that $M[S_1; S_2]_{i,j} = M_{S_1[i],S_2[j]}$.

1) Honest-to-Honest ($M^{(b)}[H; H]$).
2) From-Malicious ($M^{(b)}[\overline{H}; [n]]$).
3) Honest-to-Malicious ($M^{(b)}[H; \overline{H}]$).

The three categories are marked as A_b, B_b and C_b respectively in Figure 6.

We must enforce the messages of the From-Malicious category to be identical in both the matrices; otherwise, the attacker just checks the application messages of its peers, and knows which scenario was simulated.

In Definition 21, the $R^H_{\mathbf{N}}$ relation enforces also the Honest-to-Malicious messages to be identical, as the messages that the attacker gets are known to him (the attacker manages the application level of all the participants). The Honest-to-Honest messages are restricted by the unprotected data (The $R_{\mathbf{N}}$ part of $R^H_{\mathbf{N}}$).

In order to consider malicious destination, we change the restriction on the third messages category. Instead of enforcing the Honest-to-Malicious messages to be identical in both the matrices, we demand that rows of $M^{(1)}[H; \overline{H}]$ will be some constant permutation of $M^{(0)}[H; \overline{H}]$'s rows. The intuition for this change, is that the malicious destination should distinguish between two scenarios which differ by senders, and not by set of messages that the attacker chooses and later receives as their destination. We enforce this new restriction by defining a new relation, $R^{H,\tau}$ (see Definition 22). The relation $R^{H,\tau}_{\mathbf{N}}$ enforces all the three restrictions (see Definition 23, and Figure 6).

A. Extending the $R^H_{\mathbf{N}}$ Relation Into $R^{H,\tau}_{\mathbf{N}}$

Matrix's rows notation. For a matrix $M \in \mathcal{M}_{n\times m}(V*)$, $Rows(M)$ is the set of M's rows.

Definition 22: ($R^{H,\tau}$) For a given $n \in \mathbb{N}$, consider a pair of matrices, $(M^{(0)}, M^{(1)}) \in \mathcal{M}_{n\times n}(V*)^2$, $H \subseteq [n]$, $\overline{H} = [n] - H$ and a permutation τ over $|H|$ elements. We say that $(M^{(0)}, M^{(1)}) \in R^{H,\tau}$ if and only if

$$Rows(M^{(0)}[H; \overline{H}]) = \tau(Rows(M^{(1)}[H; \overline{H}]))$$

Definition 23: ($R^{H,\tau}_{\mathbf{N}}$) For a given $n \in \mathbb{N}$, consider a pair of matrices, $(M^{(0)}, M^{(1)}) \in \mathcal{M}_{n\times n}(V*)^2$, a relation $R_{\mathbf{N}}$ for $\mathbf{N} \in \{SUL, SA, SA*\}$, $H \subseteq [n]$ and a permutation τ over $|H|$ elements. We say that $(M^{(0)}, M^{(1)}) \in R^{H,\tau}_{\mathbf{N}}$ if and only if

1) $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}} \cap R^{H,\tau}$.

[2]www.anonymizer.com.

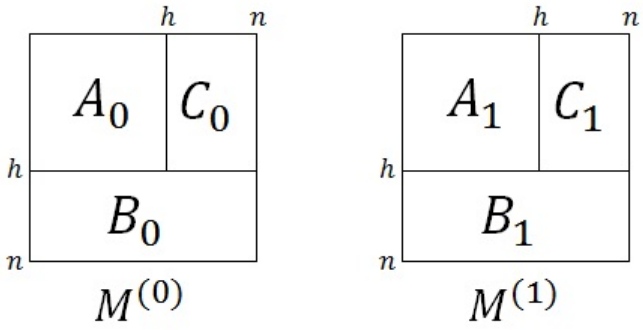

Fig. 6. Example of $R^{H,\tau}_{\mathbf{N}}$, for $H = [h] \subset [n]$. $(M^{(0)}, M^{(1)}) \in R^{H,\tau}_{\mathbf{N}}$ if and only if $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}$, $B_0 = B_1$ and $Rows(C_0) = \tau(Rows(C_1))$. In contrast to $R^H_{\mathbf{N}}$ (see Figure 5), (A_0, A_1) is **not necessarily** in $R_{\mathbf{N}} \subseteq \mathcal{M}_{|H|\times|H|}(V*)^2$.

2) For every $i \in [n] - H$ and $j \in [n]$, $M^{(0)}_{i,j} = M^{(1)}_{i,j}$.

The following lemma defines the relationship between the $R^H_{\mathbf{N}}$ and the $R^{H,\tau}_{\mathbf{N}}$ relations.

Lemma 24: For every $n \in \mathbb{N}$ and $H \subseteq [n]$, if τ is the identity permutation over $|H|$ elements then $R^H_{\mathbf{N}} = R^{H,\tau}_{\mathbf{N}}$.

Proof: Let $(M^{(0)}, M^{(1)}) \in R^H_{\mathbf{N}}$, by definition $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}} \cap R^{H,\tau}$, for τ the identity permutation. The second demand in $R^{H,\tau}_{\mathbf{N}}$'s definition is derived directly from the second demand in $R^H_{\mathbf{N}}$'s definition (see Definition 19).

Let $(M^{(0)}, M^{(1)}) \in R^{H,\tau}_{\mathbf{N}}$, by definition $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}$. Additionally, because τ is the identity permutation, for every $i \in [n]$ and $j \in [n] - H$, $M^{(0)}_{i,j} = M^{(1)}_{i,j}$ and $M^{(0)}_{j,i} = M^{(1)}_{j,i}$. ■

B. Comp-**N**-Anonymity Against Malicious Destination

We extend the definition to deal with malicious destination, by extending the capability and the Comp-**N**-b experiment (Alg 12):

1) Extending the capability of the attacker. We add a bit MD to the attacker's capability. This bit indicates whether the attacker is treated as malicious destination or not. After the addition, $Cap = (\overline{H}, EV, MD) \in \mathcal{P}([n])^2 \times \{0,1\}$. Like the other Cap's elements, we denote Cap's MD by *Cap*[MD].
   The default value of *Cap*[MD] is 0, so when testing protocol's anonymity notion not against malicious destination, the capability could be written as before the extension.
2) Extending the Comp-**N**-b experiment (Alg 12). The changes in the experiment are minor; two lines are changed slightly, and one line is added:
   a) As $\mathcal{A}$ should choose τ, we change the number of arguments returned by the $Initialize$ method in line 4:
   $< STATE_{\mathcal{A}}, \tau, 1^{rounds} > \leftarrow \mathcal{A}.Initialize(1^k, \{STATE_i\}_{i\in Cap[\overline{H}]})$.
   b) Instead of verifying that the messages matrices are in $R^H_{\mathbf{N}}$, the verification is done by $R^{H,\tau}_{\mathbf{N}}$. The change is in line 7:
   if $(M^{(0)}, M^{(1)}) \notin R^{H,\tau}_{\mathbf{N}}$ **then**.

c) Use $R_{\mathbf{N}}^{H,\tau}$ only when *Cap*[MD]=1. By Lemma 24, we can do this by adding the following line between lines 4 and 5 of Alg 12:
if $Cap[\text{MD}] = 0$ **then** $\tau = ID_{|H|}$.
Where $ID_{|H|}$ is the identity permutation over $|H|$ elements.

C. Malicious Participants vs. Malicious Destination

Although both these types of attackers can be defined as malicious participants (controlled by the attacker), we treat them differently. We explain our motivation for the separation by the following example:

Consider a P2P network where the attacker controls some malicious peers. In this network we want to ensure two different properties: Unobservability of the communication between honest peers, and sender anonymity when honest peers send messages to malicious participants. If we refer the attacker only as malicious destination, then we give the attacker unnecessary and irrelevant capability in the test of the unobservability of the communication between honest peers. On the other hand, if we refer that attacker only as malicious peer, we deal with degenerated attacker when we test sender anonymity against the destination.

APPENDIX D
EXAMPLE: UNOBSERVABLE REPORTING

A. DURP^{OR} and DURP^{Crowds} Are Comp-**UO**-Anonymous

We prove that DURP^{OR} and DURP^{Crowds} ensure unobservability against all the uninvolved parties: eavesdroppers, malicious peers (not the destination) and combinations of them.It is enough to prove Comp-**UO**-anonymity against any combination of the global eavesdropper and malicious peers. Namely, attackers with capability *Cap*=$(S \subset [n-1], [n])$.

Before presenting the reduction from the public-key encryption schema used by DURP to encrypt the messages for the destination, we recall the definition for indistinguishability under chosen-ciphertext attack (IND-CPA) based on right left oracle; we use the following definition, which is equivalent to the one in [13]:

Definition 25: Let $\psi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme, let $\mathcal{A}$ be an algorithm that has access to an oracle and returns a bit. We consider the following experiment:

$Expt^{ind-cpa-b}_{\mathcal{A},\psi}(k)$	$Oracle\ \mathcal{E}_{pk}(\mathbf{LR}(m_0, m_1, b))$
(pk, sk) $\leftarrow \mathcal{K}$	If $\lvert m_0\rvert \neq \lvert m_1\rvert$ then return ERR
$b' \leftarrow \mathcal{A}^{\mathcal{E}_{pk}(\mathbf{LR}(\cdot,\cdot,b))}(pk)$	$c \leftarrow \mathcal{E}_{pk}(m_b)$
Return b'	Return c

ψ is CPA secure if for all $\mathcal{PPT}$ algorithms, $\mathcal{A}$, there exists a negligible function, $negl$ such that,

$$|Pr[Expt^{ind-cpa-1}_{\mathcal{A},\psi}(k) = 1] - Pr[Expt^{ind-cpa-0}_{\mathcal{A},\psi}(k) = 1]|$$

$$\leq negl(k)$$

Before proving Theorem 26, we expand the Comp-**N**-b experiment to allow also $b = 2$, i.e., $b \in \{0, 1, 2\}$, and define $\Pi^H_{\mathcal{E},pk}$ for $\Pi \in \{DURP^{OR}, DURP^{Crowds}\}$.

*1) Comp-**N**-2:* The Comp-**N**-2 is almost identical to the Comp-**N**-b experiment ($b \in \{0, 1\}$, see Alg 12), but ignores the messages between the honest peers and d in both the matrices, and replace them with a constant message μ for d, the (only) valid destination in the protocol. In the Comp-**N**-2 experiment, the honest peers always want to send copies of some constant message, μ.

Formally, the Comp-**N**-2 experiment is identical to the experiment in Alg 12, but with one addition: after verifying that the matrices pair, $(M^{(0)}, M^{(1)})$, is legal, the Comp-**N**-2 experiment creates a new matrix, $M^{(2)}$ by running a transformation f_μ, on $M^{(0)}$. This is done by adding between lines 9 and 10 of the experiment: $M^{(2)} = f_\mu(M^{(0)})$. Notice that because $(M^{(0)}, M^{(1)}) \in R^H_{\mathbf{N}}$, and because honest peers send only messages to d, we could use $M^{(1)}$ as well.

The transformation $f_\mu : \mathcal{M}_{n\times n}(V^*) \rightarrow \mathcal{M}_{n\times n}(V^*)$ is defined as follows:

Given a constant message $\mu \in V$, and the honest participants set, $H = [n] - \overline{H}$, f is defined such that:

$$f_\mu(M)_{i,j} = \begin{cases} \{\mu\} & \text{if } i \in H, j = n \\ M_{i,j} & \text{otherwise} \end{cases}$$

2) $\Pi^H_{\mathcal{E},pk}$: for $\Pi \in \{DURP^{OR}, DURP^{Crowds}\}$, we define $\Pi^H_{\mathcal{E},pk}$ as follows: $\Pi^H_{\mathcal{E},pk}$ is identical to Π, such that d's public key is pk, and with one change; when a message m is taken from the application queue of a peer with index in $H \subseteq [n]$ (a message that the peer sends to the destination, or a dummy in case that the application queue is empty), or when Set is empty (see lines 7 and 10 in Alg 1), instead of encrypting it using DURP's public key encryption scheme, the encryption of m is done by $\mathcal{E}(m)$ ($\mathcal{E}(m)$ is not necessary depends on pk).

Theorem 26: Let ψ be the public-key encryption scheme used by DURP, and $l(\cdot)$ be some polynomial. If ψ is CPA-secure, then $\Pi \in \{DURP^{OR}, DURP^{Crowds}\}$ over $2 < n < l(k)$ participants is Comp-**UO**-anonymous against attackers with capability $(S \subset [n-1], [n])$.

Proof: We want to prove that for Π, n, and S as in the theorem, for all $\mathcal{PPT}$ $\mathcal{A}$ there exists a negligible function $negl$ such that:

$$Adv^{Comp-\mathbf{UO}}_{\Pi,n,\mathcal{A},(S,[n])}(k) < negl(k)$$

We prove that given a $\mathcal{PPT}$ $\mathcal{A}$, that has non-negligible advantage for some $2 < n < l(k)$ and $S \subset [n-1]$, we can build $\mathcal{A}'$ that breaks the CPA-security (Definition 25) of ψ, the public-key encryption scheme used in DURP for encrypting the messages.

We prove that for $a \in \{0, 1\}$ the attacker cannot efficiently distinguish whether it runs in the Comp-**UO**-a or in the Comp-**UO**-2 experiments while ψ is CPA-secure. Namely, for all $\mathcal{PPT}$ algorithms, $\mathcal{A}$, there exists a negligible function $negl$ such that,

$$|Pr[Expt^{Comp-\mathbf{UO}-2}_{\Pi,n,\mathcal{A},(S,[n])}(k) = 2] - Pr[Expt^{Comp-\mathbf{UO}-a}_{\Pi,n,\mathcal{A},(S,[n])}(k) = 2]|$$

$$< negl(k)$$

By the hybrid argument technique (or simply, the triangle inequality), this is enough to prove that an attacker cannot

efficiently distinguish whether it runs in the Comp-**UO**-0 or in the Comp-**UO**-1 experiments, while ψ is CPA-secure.

We now present a reduction from the CPA-security of ψ, to the indistinguishability between the Comp-**UO**-a and the Comp-**UO**-2 experiments.

Given a $\mathcal{PPT}$ adversary, $\mathcal{A}$, and some $n > 2$ and $S \subset [n]$, such that for every negligible function, $negl$:

$$|Pr[Expt^{Comp-\mathbf{UO}-2}_{\Pi,n,\mathcal{A},(S,[n])}(k) = 2] - Pr[Expt^{Comp-\mathbf{UO}-a}_{\Pi,n,\mathcal{A},(S,[n])}(k) = 2]| > negl(k) \quad (15)$$

We build a $\mathcal{PPT}$, $\mathcal{A}'$, that breaks the CPA-security of ψ, the public-key encryption scheme, used by DURP, according to Definition 25.

$\mathcal{A}'$ runs in one of the IND-CPA-b experiments without knowing whether b is 0 or 1, with an access to the encryption oracle $Oracle\ \mathcal{E}_{pk}(\mathbf{LR}(\cdot,\cdot,b))$, and it acts as follows:

1) Gets pk from the IND-CPA-b experiment.
2) $b' \leftarrow Expt^{Comp-\mathbf{UO}-a}_{\Pi^H_{\mathcal{E}_a,pk},n,\mathcal{A},(S,[n])}(k)$, such that $\mathcal{E}_a$ is defined as $Oracle\ \mathcal{E}_{pk}(\mathbf{LR}(m,\mu,b))$ for $a = 0$ and $Oracle\ \mathcal{E}_{pk}(\mathbf{LR}(\mu,m,b))$ for $a = 1$, and H is the honest participants set as defined in the third line of the experiment (see Alg 12).
3) If $b' = 2$ return 1. Otherwise return 0.

$\mathcal{A}'$ is polynomial in k as the Comp-**N**-b experiment takes $poly(k)$ time (see B-F).

We now prove the correctness of the reduction from ψ's CPA-security to the indistinguishability between the Comp-**UO**-a and the Comp-**UO**-2 experiments:

If in the IND-CPA-b experiment $b = 1 - a$, then in $Expt^{Comp-\mathbf{UO}-a}_{\Pi^H_{\mathcal{E}_a,pk},n,\mathcal{A},(S,[n])}(k)$, only encryptions of μ will be inserted into Sets of honest participants, exactly as though $Expt^{Comp-\mathbf{UO}-2}_{\Pi,n,\mathcal{A},(S,[n])}(k)$ was run. If $b = a$, then $\Pi^H_{\mathcal{E}_a,pk}$ is identical to Π. So $Expt^{Comp-\mathbf{UO}-a}_{\Pi,n,\mathcal{A},(S,[n])}(k)$ was run in the second step of $\mathcal{A}'$.

If $\mathcal{A}'$ runs in the IND-CPA-1 experiment, then its probability to return 1 is exactly the probability of $\mathcal{A}$ to return 2 in the Comp-**UO**-$(2-a)$ experiment:

$$Pr[Expt^{ind-cpa-1}_{\mathcal{A}',\psi}(k) = 1] = Pr[Expt^{Comp-\mathbf{UO}-(2-a)}_{\Pi,n,\mathcal{A},(S,[n])}(k) = 2] \quad (16)$$

In the IND-CPA-0 experiment, its probability to return 1 is exactly the probability of $\mathcal{A}$ to return 2 in the Comp-**UO**-$2a$ experiment:

$$Pr[Expt^{ind-cpa-0}_{\mathcal{A}',\psi}(k) = 1] = Pr[Expt^{Comp-\mathbf{UO}-2a}_{\Pi,n,\mathcal{A},(S,[n])}(k) = 2] \quad (17)$$

Substitution of Equations (16) and (17) in Equation (15), for $a \in \{0,1\}$, shows that ψ is not CPA-secure according to Definition 25.

Therefore, if ψ is CPA-secure, $Expt^{Comp-\mathbf{UO}-a}_{\Pi,n,\mathcal{A},(S,[n])}(k)$ and $Expt^{Comp-\mathbf{UO}-2}_{\Pi,n,\mathcal{A},(S,[n])}(k)$ are indistinguishable for $a \in \{0,1\}$. By the hybrid argument technique, we get that if ψ is CPA-secure, $Expt^{Comp-\mathbf{UO}-0}_{\Pi,n,\mathcal{A},(S,[n])}(k)$ and $Expt^{Comp-\mathbf{UO}-1}_{\Pi,n,\mathcal{A},(S,[n])}(k)$ are also indistinguishable. By definition (21), that means that Π over n participants is Comp-**UO**-Anonymous against attackers with capability $(S,[n])$. ■

REFERENCES

[1] G. Danezis and C. Diaz, "A survey of anonymous communication channels," *Computer Communications*, vol. 33, 2008.

[2] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of cryptology*, vol. 1, no. 1, 1988.

[3] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[4] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 4, pp. 482–494, 1998.

[5] M. Waidner, B. Pfitzmann *et al.*, "The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure serviceability," *J.-J. Quisquater and J. Vandewalle, editors, Advances in CryptologyEUROCRYPT*, vol. 89, p. 690, 1989.

[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*. USENIX Association, 2004.

[7] C. Shields and B. Levine, "A protocol for anonymous communication over the internet," in *Proceedings of the 7th ACM conference on Computer and communications security*. ACM, 2000, pp. 33–42.

[8] M. Rennhard and B. Plattner, "Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection," in *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*. ACM, 2002, pp. 91–102.

[9] A. Beimel and S. Dolev, "Buses for anonymous message delivery," *Journal of Cryptology*, vol. 16, no. 1, pp. 25–39, 2003.

[10] M. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 193–206.

[11] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A scalable and efficient protocol for anonymous communication," 2003.

[12] A. Hevia and D. Micciancio, "An indistinguishability-based characterization of anonymous channels," in *Privacy Enhancing Technologies*. Springer, 2008, pp. 24–43.

[13] M. Bellare and P. Rogaway, "Asymmetric Encryption," http://cseweb.ucsd.edu/~mihir/cse207/w-asym.pdf.

[14] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[15] K. Sampigethaya and R. Poovendran, "A survey on mix networks and their secure applications," *Proceedings of the IEEE*, vol. 94, no. 12, pp. 2142–2181, 2006.

[16] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The Second-Generation Onion Router," in *USENIX Security Symposium*. USENIX, 2004, pp. 303–320. [Online]. Available: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html

[17] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Advances in CryptologyCRYPTO'98*. Springer, 1998, pp. 26–45.