On the Limits of Provable Anonymity

BIU Dept. of CS, Network Security Group Technical Report 13-02 (Apr. 2013)

Nethanel Gelernter and Amir Herzberg

Department of Computer Science
Bar Ilan University
firstname.lastname@gmail.com

Abstract. We study *provably secure anonymity*, focusing on *ultimate anonymity* - strongest-possible anonymity requirements and adversaries. We begin with rigorous definition of anonymity against wide range of computationally-bounded attackers, including eavesdroppers, malicious peers, malicious destinations, and their combinations. Following [14], our definition is generic, and captures different notions of anonymity (e.g., unobservability and sender anonymity).
We then study the *feasibility of ultimate anonymity*. We show a protocol satisfying this requirement, but with absurd (although polynomial) inefficiency and overhead. We show that such inefficiency and overhead is unavoidable for 'ultimate anonymity'. We then present a slightly-relaxed requirement and present feasible protocols for it.

1 Introduction

Anonymous communication is an important goal, and is also interesting and challenging. Since the publication of the first, seminal paper by Chaum [9], there has been a large research effort by cryptography and security researchers to study anonymity and develop solutions, resulting in numerous publications and several systems. However, current work still did not address the fundamental question: what is the *strongest possible definition of anonymous communication*? By *strongest* we refer to a rigorous, indistinguishability-based definition, considering the strongest-possible adversaries and the strongest anonymity requirements; and by *possible* we refer to the feasibility of a provably-secure protocol, with polynomial resources (and, preferably, reasonable overhead and efficiency).

This paper attempts to meet this challenge. We extend the indistinguishability-based definitions of Hevia and Micciancio [14], which were limited to a benign participants, in particular benign destinations. These limitations are very significant; in fact, most of the efforts to develop and research anonymous communication, focused on anonymity against a (malicious) destination, and malicious peers are also often considered.

Our extended definitions allow adversary to control active, malicious peers (and destination). This requires us to define precise model and experiment.

Dealing with a malicious destination is esp. challenging. Indeed, many of the anonymity properties considered in the *common terminology* of Pfitzmann and Hansen [16–18], e.g., unobservability, are trivially inapplicable against a malicious destination (which can observe received traffic). We conclude, that the 'ultimate' anonymity, requires the strongest properties achievable against malicious destination, and in addition, the strongest properties achievable assuming a benign destination.

Hevia and Micciancio [14] present indistinguishability-based definitions for the different anonymity notions. Their formal definition is based on a generic experiment: the attacker chooses two scenarios and the experiment simulates one of them; the attacker should distinguish which scenario was simulated.

Another challenge we had to deal with, is that a strong adversary should be allowed to be *adaptive*. As with many cryptographic primitives, there is a significant difference between adaptive and non-adaptive adversaries (for example CCA1 and CCA2 encryption schemes [2]), and between passive and active attackers (for example security against semi honest or malicious adversaries in multi party computation protocols [13]).

To deal with adaptive and active attackers, we had to define a simulation model for the tested protocols. This challenge was not relevant or addressed in previous works [14].

Using our definitions and model, it is possible to formally prove different anonymity notions with respect to different attacker capabilities. We define the capability of the attacker by the protocol's participants it controls, and the participants to whom it can eavesdrop. Protocols can have different anonymity notions against different attackers with different capabilities.

1.1 Contributions

Our main contribution is in presenting rigorous, indistinguishability-based definitions for anonymous communication protocols, whose anonymity is assured even against strong, malicious, adaptive attackers, which may control nodes (Sections 2), possibly including the destination (Section 3). Previous rigorous definitions [14] were limited to eavesdropping attackers, not even ensuring anonymity against the destination; therefore, this is significant, critical extension.

We actually explore two variants of this definition. The stronger requirements essentially formalizes the strongest anonymity considered in the literature, e.g., in the common terminology [16–18]. We show that it is possible to achieve this variant, albeit, with an inefficient protocol (more a 'proof of feasibility' than a real protocol). However, we further show, that this inefficiency is unavoidable, i.e., we prove that *any* protocol meeting this variant of the definition, would be very inefficient (Section 5).

This motivates slightly relaxing the anonymity requirements, as we do in our second definition (Section 6). Indeed, we show that this slightly-relaxed definition can be satisfied, with reasonable efficient protocols. For example, the classical DC-net protocol [8] that fails to satisfy the stronger requirement, does satisfy this slightly weaker requirement. In the full version, we also present improved

protocols, which ensure this anonymity property even against multiple malicious nodes.

1.2 Related Works

There is a huge body of research in theory and practice of anonymous communication, beginning with Chaum's paper [9]; see, e.g., a survey of known protocols in [20]. Even just focusing on the closely related works, focusing on rigorous definitions, would far exceed the length limitations, and is delegated to the full version. A good overview of related rigorous works appears in [14], where Hevia and Micciancio presented rigorous, indistinguishability-based definitions to most anonymity notions, limited to passive, non-destination adversaries. Our work extends [14] to deal with strong, active, malicious attackers, including destination.

Few recent works extend [14] in different ways, e.g., applying the UC framework [7] for anonymous communication [24], and further studying relations among the notions [6, 15]. However, these works do not address our goals of studying the strongest anonymity notions (against strongest adversaries).

The latest version of the common terminology [17] contains comparison between the terminology to the anonymity notions in [14].

A framework for formalizing and comparing identity-related properties is offered in [25], however, differently from our approach, they ignore the confidentially of the messages content, and therefore cannot capture many of the anonymity notions our definition captures (e.g., see section 6.2 there).

2 Definitions

Following Hevia and Micciancio [14], 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.

2.1 Network model, adversary and peers

Since our goal is to study anonymity against adaptive and active attackers, we need a rigorous communication and execution model. In this work, we adopt the simplest model: fully synchronous ('rounds/iterations') communication with instantaneous computation, allowing direct communication between every two participants (clique).

Peers. We let the adversary control the 'application layer' of all peers, i.e., deliver requests to the protocol layer, to send messages to particular destination(s). In the protocol layer, the honest peers follow the protocol and are simulated by the experiment, while the attacker controls the 'malicious peers'.

Different peers can have different roles in the protocol; for example, protocols that use mixes [9,21] or routers [10] to assist anonymous communication by other peers, often have two types of peers: client and mix (or router). The roles of the participants are determined by the protocol.

Adversary. The attacker controls the application layer of all the peers. Namely, the attacker chooses, for every peer and at every round, a matrix of messages (from each peer, and to each peer). The anonymity requirements are defined by an indistinguishability game, where the attacker selects two sets of matrices (for each round) and the game selects one of them, and the adversary tries to detect which of the two sets was selected. Different anonymity notions, are represented by different restrictions on the matrices. In peers that it controls, the attacker can also deviate arbitrarily from the protocol (i.e., act in a malicious/byzantine manner).

The power to select the entire sequence of messages to be sent (the matrices) might seem excessive. However, this follows the same 'conservative' approach applied in experiments of cryptographic primitives such as encryption [4] [2]. As mentioned in [14], in reality, the attacker might have some 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.

2.2 Experiment: parameters, notations and security notions

Notations We use the following common cryptographic and mathematical notations: $\mathcal{PPT}$ is the set of probabilistic polynomial time algorithms. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, ..., n\}$. We use $\mathcal{P}(S)$ to denote the power set of set S. Namely, consider two sets, S_1 and S_2; then $S_1 \in S_2{}^*$ if and only for every $s \in S_1$, $s \in S_2$.

We use $V = \{0,1\}^l$ to denote the messages space. A collection of messages between n parties is represented by an $n \times n$ matrix, $M = (m_{i,j})_{i,j \in [n]}$. Each element $m_{i,j} \in V^*$ is the multiset of messages from the i-th party to the j-th party [1].

The experiment parameters: ($\pi, n, \mathcal{A}$ and *Cap*. 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. To initialize the parties, e.g., establish shared (or

[1] We replaced [14]'s notation $\mathcal{P}(V)$ with V^*, because a powerset does not contains multisets

public/private) keys, we use π's *setup* method, which receives the number of participants n and the identity i of a specific participant as parameter, and outputs the initial state of i (denoted by $STATE_i$); this follows the 'common reference string' model. In practice, ths simply means that we assume the parties have appropriate keys (shared or public/private). The π's *simulate* method receives the current state of a participant, together with its incoming traffic and new messages from the application layer, and returns its next state and its outgoing traffic and messages to deliver to the application.

The two final experiment's parameters, $\mathcal{A}$ and *Cap*, define the attacker. $\mathcal{A}$ is the attacker $\mathcal{PPT}$ algorithm. The *Cap* parameter defines the attacker capabilities, and consists of two sub-parameters, i.e., $Cap \in \mathcal{P}([n])^2$, which we denote $Cap = (\overline{H}, EV)$. The $\overline{H}$ parameter specifies the machines controlled by adversary $\mathcal{A}$, and the EV parameter identifies machines to which the attacker can eavesdrop (e.g., all machines, for a global eavesdropper). To refer to a specific parameter of *Cap* we use the notation $Cap[\overline{H}]$ and $Cap[EV]$ respectively. An attacker with capability $Cap = (\overline{H}, EV)$, controls the machines with indexes in $Cap[\overline{H}]$ and eavesdrops the traffic of the machines with indexes in $Cap[EV]$.

In the next section, we extend the capability to deal also with malicious destination, by adding to *Cap* another bit, $Cap[MD]$; the definition of this section is the same as that definition, using $MD = 0$.

Security notions The unprotected data (the attacker assumed knowledge), is defined by relations on matrices, $R_{\mathbf{N}} \subseteq \mathcal{M}_{n\times n}(V^*)^2$. Definitions of all anonymity notions appears in [14]; while all of these are applicable to our experiment, we focus on the strongest relations could be achieved against the different attackers: *unobservability* (UO) and *sender anonymity* (SA).

The relevant relations are $R_{\mathbf{UO}}$ and $R_{\mathbf{SA}}$. The unobservability relation $R_{\mathbf{UO}}$ simply holds for all matrix pairs, i.e., does not restrict the matrices at all. The sender anonymity relation $R_{\mathbf{SA}}$ requires that for every (recipient) i, in both the matrices the i-th column contains the same messages. Namely, every participant receives the same messages. That way, the attacker can distinguish between the scenarios only by the senders.

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

Definition 1. *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 $(M^{(0)}, M^{(1)}) \in R^H_{\mathbf{N}}$ if and only if*

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

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

The case when messages are sent from honest peers to corrupted, is the case of malicious destination that is discussed in Section 3. $R^H_{\mathbf{N}}$ extends the demand of identical unprotected data in both the matrices, to active attackers. Figure 1 depicts the relation.

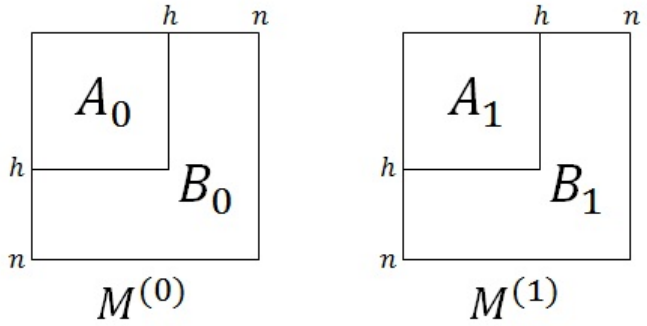

Fig. 1: Example of $R^H_{\mathbf{N}}$, for $H = [h] \subset [n]$. $(M^{(0)}, M^{(1)}) \in R^H_{\mathbf{N}}$ if and only if $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}$ and B_0 and B_1 contain only empty messages multisets. Notice that $(A_0, A_1) \in R_{\mathbf{N}} \subseteq \mathcal{M}_{|H|\times|H|}(V^*)^2$.

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.

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

The experiment (see Algorithm 1) 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, $\mathbf{N}$ (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.

Algorithm 1 $Expt^{\mathbf{Comp-N-b}}_{\pi,n,\mathcal{A},Cap}(k)$

for $i = 1$ **to** n **do** $STATE_i \leftarrow \pi.Setup(1^k, i, n)$ **end for**
$EV = Cap[\overline{H}] \cup Cap[\mathrm{EV}]$
$H = [n] - Cap[\overline{H}]$
$< STATE_{\mathcal{A}}, 1^{rounds} > \leftarrow \mathcal{A}.Initialize(1^k, \{STATE_i\}_{i \in Cap[\overline{H}]})$
for $t = 1$ to $rounds$ **do**
  $< STATE_{\mathcal{A}}, M^{(0)}, M^{(1)} > \leftarrow \mathcal{A}.InsertMessages(1^k, STATE_{\mathcal{A}})$
  if $(M^{(0)}, M^{(1)}) \notin R^H_{\mathbf{N}}$ **then**
    return 0
  end if
  for each $i \in H$ **do**
    $< STATE_i, \{C_{i,j,t}\}_{j=1}^n > \leftarrow \pi.Simulate(1^k, STATE_i, \{C_{j,i,t-1}\}_{j=1}^n, \{m^{(b)}_{i,j}\}_{j=1}^n)$
  end for
  $< STATE_{\mathcal{A}}, \{C_{i,j,t}\}_{\substack{i \in Cap[\overline{H}] \\ 1 \le j \le n}} > \leftarrow \mathcal{A}.Simulate(1^k, STATE_{\mathcal{A}}, \{C_{i,j,t-1}\}_{i \vee j \in EV})$
  $< STATE_{\mathcal{A}}, b' > \leftarrow A.GuessB(1^k, STATE_{\mathcal{A}})$
  if $b' \neq NULL$ **return** b' **end if**
end for
return 0

Definition 2. *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 3. *Protocol π is Comp-$\mathbf{N}$-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,

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

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

The total runtime of the experiment (Alg 1) is critical for the proof of the anonymity notions. Using our definition, it is possible to formally prove anonymity notions by a polynomial time reduction to 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 does 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.

3 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 of messages from honest peers. 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 [10] and services like Anonymizer [2].

In this section we extend Definition 3 to deal also with malicious destination. This extension is relevant only for two Comp-**N**-anonymity notions: $\mathbf{N} \in \{SUL, SA\}$ (SUL for sender unlinkability [14]). The other anonymity notions are aimed to hide information that the destination has, and therefore they are irrelevant in such an attacker model.

To extend the definition also against malicious destination, we apply the $R_{\mathbf{N}}$ relation also on the messages from honest peers to malicious. We enforce this new restriction by defining a new relation, $\widehat{R}^H_{\mathbf{N}}$. Figure 2 depicts the new relation.

Definition 4. *($\widehat{R}^H_{\mathbf{N}}$) 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 $\mathbf{N} \in \{SUL, SA\}$, and $H \subseteq [n]$. We say that $(M^{(0)}, M^{(1)}) \in \widehat{R}^H_{\mathbf{N}}$ if and only if*

[2] www.anonymizer.com.

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

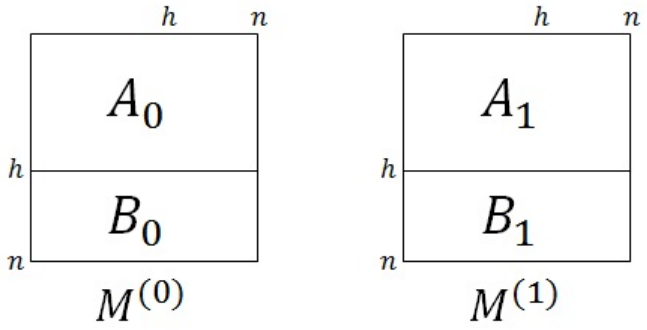

Fig. 2: Example of $\widehat{R}^{H}_{\mathbf{N}}$, for $H = [h] \subset [n]$. $(M^{(0)}, M^{(1)}) \in R^{H,\tau}_{\mathbf{N}}$ if and only if $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}$, and B_0 and B_1 contain only empty messages multisets.

3.1 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 1).

Extending the attacker's capability 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.

Extending the Comp-N-*b* experiment (Alg 1) . The messages matrices verification should be done either by $R^{H}_{\mathbf{N}}$ or by $\widehat{R}^{H}_{\mathbf{N}}$, according to the attacker capability. The change is in line 7:

if (Cap[MD] $= 0$ **and** $(M^{(0)}, M^{(1)}) \notin R^{H}_{\mathbf{N}}$) **or** $(M^{(0)}, M^{(1)}) \notin \widehat{R}^{H}_{\mathbf{N}}$ **then**.

4 Comp-N-Anonymity Discussion

In this section we discuss the definition of the previous section. We begin with comparison of our definition and the definition of Hevia and Micciancio [14],

and then we discuss the Comp-**N**-anonymity of anonymity protocols that rely on high traffic from many users. In the end of this section, we discuss the relations between the different Comp-**N**-anonymity notions, and the anonymity of Comp-**N**-anonymous protocols against attackers with capability Cap, against attackers with other capabilities.

4.1 Our Definition vs. [14]'s Definition

The most striking difference between the definitions, is the kinds of the attackers. While [14] deals only with a passive global eavesdropper, our definition take into consideration wider range of attacker.

Another difference is that in contrast to [14], our definition gives the attacker to control the application level adaptively. In [14]'s experiment, only once, at the beginning of the experiment, the adversary chooses both the matrices. Then the protocol is simulated by one of the matrices, and the adversary should guess which one was simulated. An example that illustrates the difference appears in Appendix C.

[14]'s definition in our model We now present a short and simple change to our experiment for making [14]'s definition a special case of our definition. The change is additional bit to the attacker's capability, that indicates whether the attacker controls the application adaptively or not. After the addition, $Cap \in \mathcal{P}([n])^2 \times \{0,1\}^2$. If the bit in the capability is 0 (the attacker is non-adaptive), than the experiment gives the attacker to choose the matrices $M^{(0)}$ and $M^{(1)}$ only in the first iteration, and in the rest of the iterations, the experiment fixes the matrices to be empty messages matrices.

In their experiment, Hevia and Micciancio do not specify how to simulate the protocol, but under our protocol's simulation model (see Alg 1) with the new addition, we can test their definition. In their model the attacker is a passive global eavesdropper that does not control the application adaptively; so, we just specify the attacker's capability to be ($\emptyset$, $[n]$, 0, 0). π's setup method of the simulated protocol, returns initial states of n machines.

4.2 Traffic Based Anonymity

The definition of Comp-**N**-Anonymity, gives the attacker the power to control the application level of all the protocol's participants. Therefore, anonymity protocols that their anonymity mainly depends on high traffic from many users, are usually not Comp-**N**-anonymous, even for the weaker **N**-anonymity notions, and against weak attackers.

In Appendix D, we bring a detailed example of a simplified version of the Tor protocol [10]. Tor is the most popular anonymity network ever, and it provides anonymity for millions of users. We show that Tor is not Comp-**N**-Anonymous, for any **N** (see Table **??**), even against the weakest attacker: a local eavesdropper to one arbitrary Tor router.

4.3 Relations Between the Comp-N-Anonymity Properties

Relations between Comp-N-anonymity notions Similarly to [14], some Comp-**N**-anonymity notions imply other, against attackers with the same capability: UO $\rightarrow$ SRA $\rightarrow$ SA* $\rightarrow$ SA $\rightarrow$ SUL, SRA $\rightarrow$ RA* $\rightarrow$ RA $\rightarrow$ RUL, SA* $\rightarrow$ UL $\rightarrow$ RUL and RA* $\rightarrow$ UL $\rightarrow$ SUL. This stems directly from the definition of the $R^H_{\mathbf{N}}$ relation; for every attacker's capability Cap, for every relation above of the form X $\rightarrow$ Y , $R^H_{\mathbf{Y}} \subset R^H_{\mathbf{X}}$. Hence, an attacker $\mathcal{A}$ that has non-negligible advantage $Adv^{Comp-\mathbf{Y}}_{\pi,n,\mathcal{A},Cap}(k) > negl(k)$, also have non-negligible advantage for the Comp-**X**-anonymity notion.

Relations between the different attacker's capabilities We first show that as expected, enlarging $Cap[EV]$, the attacker's eavesdropped participants set, can only increase the attacker's power.

Lemma 1. *If π is a Comp-**N**-anonymous protocol (for some anonymity notion **N** [14]) against attackers with capability Cap = ($\overline{H}$, EV, MD), then π is also Comp-**N**-anonymous against attackers with capability Cap' = ($\overline{H}$, EV' $\subset$ EV, MD).*

Proof. (Sketch) We briefly show how to build an attacker $\mathcal{A}$ with capability Cap that breaks π's Comp-**N**-anonymity, given an attacker $\mathcal{A}'$ with capability Cap' that can do that.

$\mathcal{A}$ runs in one of the Comp-**N**-b experiments ($b \in \{0,1\}$). It then simulates $\mathcal{A}'$ over the experiment, and does exactly what $\mathcal{A}'$ does. $\mathcal{A}$ that gets information according to Cap, pass information to $\mathcal{A}'$, only according to Cap'. The correctness of the reduction is trivial. □

In contrast, enlarging $Cap[\overline{H}]$, the attacker's controlled participants set, might detract its power. The intuition for the counter-example we bring here, is that controlling participants has some price; the $R^H_{\mathbf{N}}$ and $\widehat{R}^H_{\mathbf{N}}$ relations, restrict the controlled participants ($Cap[\overline{H}]$).

Lemma 2. *If π is a Comp-**N**-anonymous protocol (for some anonymity notion **N** [14]) against attackers with capability Cap = ($\overline{H}$, EV, MD), then π **not necessarily** Comp-**N**-anonymous against attackers with capability Cap' = ($\overline{H}'$ $\subset$ $\overline{H}$, EV, MD).*

Proof. (Sketch) Every protocol is Comp-**N**-Anonymous against attacker that controls all the participants ($\overline{H} = [n]$), because in such a case, there is no protected data at all (from the attacker), and both the messages matrices must be identical. But, there are protocols that are not Comp-**N**-anonymous even against *one* controlled participant. The example in Appendix D, when the attacker controls one router, instead of eavesdropping to it, demonstrates this (for every **N**). □

The last lemma shows that for $\mathbf{N} \in \{SUL, SA\}$, extending the capability of an attacker to be also malicious destination, does not detract the attacker's power.

Lemma 3. *For* $\mathbf{N} \in \{SUL, SA\}$*, if* π *is a Comp-*$\mathbf{N}$*-anonymous protocol against attackers with capability Cap =* $(\overline{H}, EV, 1)$*, then* π *is also Comp-*$\mathbf{N}$*-anonymous against attackers with capability Cap' =* $(\overline{H}, EV, 0)$*.*

Proof. Directly from the definitions of the $R^H_{\mathbf{N}}$ and $\widehat{R}^H_{\mathbf{N}}$ relations: for every $\mathbf{N} \in \{SUL, SA\}$ and $H \subseteq [n]$, $R^H_{\mathbf{N}} \subseteq \widehat{R}^H_{\mathbf{N}}$. □

5 Feasibility of Comp-N-Anonymity Against Strong Attackers

We say that protocol ensures *ultimate anonymity* if it ensures both the strongest anonymity notions we can require from a protocol:

1. Comp-**SA**-anonymity (sender anonymity) against malicious destination that is a global eavesdropper and controls additional participants (in short: *strong malicious destination*).
2. Comp-**UO**-anonymity (unobservability) against global eavesdropper and malicious peers (*strong attacking peers*).

In order to exclude the trivial solution of the protocol that does not send any message, we limit the discussion to protocols that ensure the *liveness* property; informally, protocols that while the attacker do not deviate from the protocol ensures messages delivery.

Stronger property we would like to get is t*-liveness*; informally, a protocol satisfies t-liveness if it ensures messages delivery in the presence of up to t malicious participants.

While ensuring unobservability against strong attacking peers is almost trivial, it is complicated to ensure sender anonymity against strong malicious destination and both the demands together (ultimate anonymity).

None of the known protocols [20] satisfies ultimate anonymity. This is not surprising, because in the end of the section, we prove that the cost of ultimate anonymity is very inefficiency (Theorem 3). However, we prove that there exists a protocol that ensures ultimate anonymity. This protocol satisfies also t-liveness for $t < \frac{n}{2}$ (Theorem 2).

In the next section, we present a relaxed definition to Comp-**N**-anonymity against malicious destination that can be satisfied much more efficiently.

5.1 Ensuring Comp-UO-Anonymity Against Strong Attacking peers

It is possible to ensure any anonymity notion **N**, against any combination of malicious participants and eavesdroppers (without malicious destination).

We now present a simple protocol that ensures Comp-**UO**-anonymity against strong attacking peers (and therefore ensures all the other anonymity notions; see Section 4.3). In this protocol, every round, every participant sends a message (real or dummy) to every other participant; the communication is semantically secure encrypted [4].

As in this protocol honest peers communicate with each other directly, no information can be learned about the scenarios without learning information from the encrypted content. Formal proof of this protocol could be done by reducing the Comp-**UO**-anonymity to the security of the encryption scheme.

Protocols that ensure Comp-**UO**-anonymity and yet provides some anonymity (although not Comp-**SA**-anonymity; see below) are DC-net [8] and mixnet [9] based protocols [21] that use constant rate sending.

5.2 Known Protocols are Not Comp-SA-Anonymous Against Strong Malicious Destination

None of the known protocols [20] is a Comp-**SA**-anonymous protocol against strong malicious destination (especially when the attacker controls any minority of the participants). In Theorem 2 we show that such a protocol exists; Theorem 3 shows that any Comp-**SA**-anonymous protocol against malicious destination that is also global eavesdropper, must have high overhead.

As example to hardness of the demand, we bring the DC-net protocol [8] that ensures sender anonymity against the destination by every definition we encountered. We disprove DC-net's Comp-**SA**-anonymity even against passive destination alone in Appendix B.2. Briefly, DC-net fails to hide whether two messages are sent by the same peer or by different two peers.

Similar attack works also against a scheme of many peers that sends via mix or mixes cascade [9]. When the destination controls also some mixes, other attacks are possible [22].

5.3 Ensuring Ultimate Anonymity Against Strong Attackers

It was shown that it is feasible for n parties to compute a polynomially-computable functionality f, without any of them learning anything but the result of the computation, even if some minority of them are malicious. This is possible using techniques of secure multi-party computation [13]. There has been many results in this area, where the most basic ones are of BGW [5, Theorem 3] [1] with malicious minority of less than $\frac{n}{3}$, and of GMW [12] with any malicious minority. In Theorem 2 we prove that there exists a protocol that satisfies both ultimate anonymity and t-liveness for every $t < \frac{n}{2}$. The proof relies on the GMW's theorem [12] (informally in Theorem 1) although for the purpose of feasibility proof, other protocols would be useful as well.

Theorem 1. *(*Informal): *Consider a synchronous network with pairwise semantically secure encrypted channels. Then:*

For every polynomially-computable n-ary functionality f, there exists a polynomial time protocol for computing f with computational security in the presence of a malicious adversary corrupting up to $\frac{n}{2}$ of the parties. Namely, every party learns no more than its own inputs and outputs.

Theorem 2. *There exists protocol Π such that that:*

1. *For every $t < \frac{n}{2}$, Π satisfies t-liveness.*
2. *Π ensures ultimate anonymity; i.e., for every $S \subset [n]$, $|S| < t$, and every $\mathcal{PPT}$ attacker $\mathcal{A}$, $Adv^{Comp-\mathbf{SA}}_{\Pi,n,\mathcal{A},(S,[n],1)}(k)$ and $Adv^{Comp-\mathbf{UO}}_{\Pi,n,\mathcal{A},(S,[n],0)}(k)$ are negligible.*

Proof. We presents n-ary functionality that given a trusted party, satisfies both the anonymity demands of ultimate anonymity and t-liveness. Relying Theorem 1, this trusted party can be replaced with the n participants such that $t < \frac{n}{2}$ of them are malicious. The functionality is aimed to send up to some $S \in poly(k)$ messages.

For simplicity, we consider a simple scheme of n participants, such that all the participants send anonymous messages only to one of them (the destination).

The n-functionality f is described in Algorithm 2. As an input to the secure computation, every peer chooses the lexicographically-first message in its application queue (or a dummy message if the application queue is empty), and as output the destination receives the lexicographically-lowest message, and the other peers receives empty output.

f has a state that saves all the real messages lexicographically-sorted; we refer this state as a priority queue by lexicographic order, PQ. Because the state must be of constant size (otherwise, the attacker can learn about the number of real messages that were sent), we choose $|PQ| = S$, and to prevent learning from overflows, we limit the number of the delivered messages by the protocol to S (we could do better, but for the feasibility proof S is enough). f is polynomially-computable in the security parameter k (the inputs length and $|PQ|$ are $\in poly(k)$, and the f is polynomial time algorithm).

Our experiment is synchronous, and pairwise semantically secure encrypted channels can be ensured during the setup stage of the experiment (Alg 1). Therefore, from Theorem 1 it is enough to prove that a protocol with trusted party that gets n inputs satisfies both the theorem requirements, when some minority of the inputs is completely controlled by the attacker, and the other (the ones that represent the honest peers) are restricted by the relevant relations.

Comp-SA-anonymity against strong malicious destination We prove that given any $S \subset [n]$, $|S| \leq t < \frac{n}{2}$, and a trusted party that calculates the n-ary functionality f (see Alg 2), and that the communication between the trusted party and the peers is secure, it holds that $Adv^{Comp-\mathbf{SA}}_{\Pi,n,\mathcal{A},(S,[n],1)}(k) \leq negl(k)$.

Namely, given n peers, some minority of them is malicious, a trusted party, and a global eavesdropper malicious destination that controls the malicious peers, no $\mathcal{PPT}$ attacker $\mathcal{A}$ can distinguish between any two scenarios with identical unprotected data with non-negligible probability.

In the proof we assume the destination of the messages is one of the malicious peers; otherwise, the proof is as for the unobservability case.

The only information that the attacker receives is the outputs to the (only) malicious destination from the trusted party. We prove that in every two scenarios with the same unprotected data, i.e., two scenarios that are represented

Algorithm 2 The n-ary functionality f.

State: A priority queue by lexicographic order PQ, and *Counter* for the incoming real messages. The initial state of PQ is an empty priority queue, and *Counter* starts from 0.

Input: n messages $(m_1, m_2, ..., m_n)$.

Output: We denote the destination as the i-th party; the output is $(o_1, o_2, ..., o_i, ..., o_n)$, such that o_i is the first message in the priority queue PQ (or $\perp$ message if PQ is empty), and for every $j \neq i$, $o_j = \perp$.

1: **for each** message m in $Sort(m_1, m_2, ..., m_n)$ **do**
2: **if** m is a real message **and** $Counter < |PQ|$ **then**
3: $PQ.insert(m)$
4: $Counter = Counter+1$
5: **end if**
6: **end for**
7: **if** PQ is empty **then**
8: $m = dummy$
9: **else**
10: $m = PQ.removeHead()$
11: **end if**
12: $Output = (\perp)^n$
13: $Output[i] = m$
14: **return** Output

by two sequences of messages matrices $\{M_i^{(0)}\}_{i=1}^s$ and $\{M_i^{(1)}\}_{i=1}^s$, such that for every $1 \leq i \leq s$, $(M_i^{(0)}, M_i^{(1)}) \in \widehat{R}_{\mathbf{SA}}^H$, the information that the attacker gets is identical in both the scenarios.

Every round of the protocol, the malicious destination receives one message. We claim that in both the scenarios, it receives exactly the same messages in the same order. Every honest peer sends the lexicographically-first message from its application level that has not sent yet, to the trusted party. Malicious peers might sends whatever they want. Among all these messages, the trusted party sends to the destination the lexicographically-first message. Therefore every round the message with the lowest lexicographic value (from all the application messages that have not reached the destination until this round) is sent to the destination. This happens regardless the distribution of the application messages among the l senders, because the lexicographically-first message among the messages from the honest peers' application level is always sent to the trusted party.

In both the scenarios, due to $\widehat{R}_{\mathbf{SA}}^H$, every round the same messages are inserted into the application queue of the honest peers for the destination (the only possible difference is the distribution of the messages among the honest potential sender). The peer with the lexicographically-first message in each scenario will send it to the trusted party.

Because the attacker receives identical information in both the scenarios, it cannot distinguish between the scenarios.

Comp-UO-anonymity against strong attacking peers Similarly to the Comp-**SA**-anonymity proof, and under the same notions, we need to prove that $Adv^{Comp-\mathbf{UO}}_{\Pi,n,\mathcal{A},(S,[n],1)}(k)$ is not negligible in k for any $\mathcal{A}$.

We assume the attacker does not control the destination; otherwise it is impossible to ensure unobservability. From Theorem 1, the malicious peers cannot learn more than their own inputs and outputs. But their inputs are chosen regardless of the honest peers inputs, and by f (Alg 2) the malicious peers' output is always $\perp$. Consequently, the attacker does not learn any information about the simulated scenario.

***t*-liveness for $t < \frac{n}{2}$** According Theorem 1, Π ensures delivery of S messages while only some minority of the participants is malicious. We now prove that any honest peer, can ensure message delivery of his own $\lfloor \frac{S}{n} \rfloor$ messages. Π ensures the delivery of the first S messages. Every round, every peer can add one messages to PQ, therefore in the first $\lfloor \frac{S}{n} \rfloor$ rounds every peer can add $\lfloor \frac{S}{n} \rfloor$ messages.

The attacker can limit the honest peers to this number of messages (the relative share of the peer), and can only affect the delay by sending the lexicographically-lowest messages. □

Remarks on the protocol Π (described in the proof)

1. Theorem 3 shows that the throughput of any protocol that satisfies ultimate anonymity cannot be higher than the minimal sending rate of some sender.
2. The lexicographic order of the messages in the application queues and in PQ is necessary. Let Π' be identical protocol, but such that the messages are chosen uniformly out of the application queues, and the trusted party's priority queue (by lexicographic order) is replaced with a multiset of messages, such that the message to send is chosen uniformly among the messages in the multiset.
   We consider the following two scenario: In the first scenario, only p_1 sends the destination $\{m_1, m_1, m_1, m_2\}$, and in the second scenario p_1 sends the destination $\{m_1, m_1, m_1\}$, and p_2 sends the additional m_2. Malicious destination attacker can distinguish between the scenarios by the distribution of the first message that arrives. In the first scenario, the probability of m_2 to reach the destination first is $\frac{1}{4}$, while in the second scenario, the probability of the same event is $\frac{1}{2}$. Consequently, Π' is not Comp-**SA**-anonymous against malicious destination.

5.4 Malicious Destination and Inefficiency

We now prove that the cost of ensuring Comp-**SA**-anonymity against strong malicious destination must be low efficiency (high communication overhead).

We define the number of messages that a peer p_i sends in the first R rounds of a run (scenario) σ of some deterministic protocol by $L_i(\sigma, R)$.

Theorem 3. *For every deterministic protocol, π, if π is Comp-***SA***-anonymous against malicious destination that is also global eavesdropper, then for every run (scenario) of π, σ and $R \in \mathbb{N}$, during the first R rounds of σ:*

1. *The maximal number of messages that reach the destination is $MaxOut_{\sigma,R} = min\{L_i(\sigma, R) | p_i$ is a honest potential sender$\}$.*
2. *The minimal number of messages that were sent during the first R rounds of σ is $ComOver_{\sigma,R} \geq MaxOut_{\sigma,R} \cdot |\{p_i$ is a honest potential sender$\}|$.*

A proof for the theorem appears in Appendix A. Similar theorem for probabilistic protocols will appear in the full paper. An important observation that follows the above theorem, is that when the peers send independently of each other (must happen in the case of malicious peers), because the maximal output is bounded, the number of the messages in the protocol level is on increase (and therefore also the used storage).

The above theorem is for protocols that partially satisfies the first demand of ultimate anonymity. Against attackers that satisfies the first demand, and for protocols that satisfies ultimate anonymity, the values of efficiency metrics like maximal output ($MaxOut$), communication overhead ($ComOver$) and latency, are worse. We will formally state and prove the above observations in the full paper.

6 Indistinguishability Between Permuted Scenarios

The Comp-**SA**-anonymity definition against malicious destination (see Section 3) is very hard and expensive (Theorem 3) to achieve, and therefore also ultimate anonymity. The power of malicious destination attacker might seems extremely strong: the attacker chooses the messages to send, affects their timing, and in addition is able to receive these messages and learn information from their arrival times.

This motivates us to create relaxed definition to anonymity notions against malicious destination. Like the extension to the definition in Section 3, this extension is relevant only for two anonymity notions: $\mathbf{N} \in \{SUL, SA\}$.

6.1 Permuted Scenarios

In this subsection we present a relaxed relation between the matrices of the messages sent in the two scenarios. We add a restriction on the two chosen scenarios: the only difference between them should be the identities of the senders. We enforce this new restriction, by verifying that for every pair of messages matrices (chosen by the attacker), the rows of the first matrix are some constant permutation of the other.

The same permutation must be used during the whole experiment (we give the attacker to choose it), otherwise some of the problems of the extension in Section 3 arise again. We enforce this new restriction by defining a new relation $R_{\mathbf{N}}^{H,\tau}$ (Definition 5).

Matrix's rows notation. For a matrix $M \in \mathcal{M}_{n\times m}(V^*)$, and for $H \subseteq [n]$, $Rows(M)[H]$ is the set of M's rows with indexes $\in H$.

Definition 5. *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 $\mathbf{N} \in \{SUL, SA\}$, $H \subseteq [n]$ and a permutation τ over $|H|$ elements. We say that $(M^{(0)}, M^{(1)}) \in R_{\mathbf{N}}^{H,\tau}$ if and only if*

1. $(M^{(0)}, M^{(1)}) \in \widehat{R}_{\mathbf{N}}^{H}$.
2. $Rows(M^{(0)})[H] = \tau(Rows(M^{(1)})[H])$.

6.2 Comp-$\widehat{\mathbf{N}}$-Anonymity Against Malicious Destination

We denoted the relaxed anonymity notions by $\widehat{\mathbf{N}}$. *Relaxed ultimate anonymity* is ultimate anonymity (see Section 5), but with Comp-$\widehat{\mathbf{SA}}$-anonymity instead of Comp-**SA**-Anonymity. We extend the definition to deal with malicious destination, almost as described in Section 3.1, i.e., the capability is extended, and in the experiment (Alg 1), if $Cap[MD]$=1, the matrix verification in line 7 is done by $R_{\mathbf{N}}^{H,\tau}$ instead of $\widehat{R}_{\mathbf{N}}^{H}$.

Additionally, as $\mathcal{A}$ should choose τ, we change the number of arguments returned by the *Initialize* method in line 4:

$< STATE_{\mathcal{A}}, \tau, 1^{rounds} > \leftarrow \mathcal{A}.Initialize(1^k, \{STATE_i\}_{i\in Cap[\overline{H}]})$.

6.3 Feasibility of the Permuted Comp-$\widehat{\mathbf{SA}}$-anonymity

Under the new extension, the DC-net protocol [8] in a ring topology, ensures also Comp-$\widehat{\mathbf{SA}}$-anonymity even against malicious destination that is also a global eavesdropper that controls another malicious destination (see Appendix B.3). In more complex topologies, DC-net ensures anonymity even against higher number of malicious peers [26] [11]. In spite of that, DC-net does not ensures t-liveness. In Appendix E, we present a protocol with communication overhead $O(t^3)$ that ensures relaxed ultimate anonymity when the attacker controls $t < \sqrt{n}$ participants, and also satisfies t-liveness.

7 Conclusions and Directions

We presented modular definitions covering multiple anonymity notions, against a variety of attackers: eavesdroppers, malicious peers, malicious destination and combinations of them.

None of the known protocols [20] satisfies *ultimate anonymity*, i.e., sender anonymity against strong malicious destination *and* unobservability against strong attacking peers; this motivates our study of the feasibility of ultimate anonymity. We proved that there exist a protocol that satisfies ultimate anonymity and also ensures messages delivery, when the attacker controls a minority of the participants. We then showed that any protocol that satisfies the definition against

strong malicious destination attacker, must be inefficient. Motivated by this theorem, we offered relaxed definition to anonymity notions against the destination. Some known protocols like DC-net [8] satisfy relaxed ultimate anonymity.

The first challenge that comes following our work, is to explore the space between protocols that fail to satisfy the ultimate anonymity, and the extremely inefficient protocol (although polynomial) that satisfies it. Namely, to find more efficient protocols that satisfy ultimate anonymity, and better bounds for the efficiency metrics of them. The second challenge is to find the most efficient protocols that ensure relaxed ultimate anonymity, esp., together with robustness requirements. Another interesting direction is to find bounds for the communication overhead of protocols that satisfy anonymity notions with regarding to the t-liveness property they satisfy. Finally, it would be interesting to explore the implications of relaxing the model, e.g., removing the synchronization assumptions.

References

1. Gilad Asharov and Yehuda Lindell. A full proof of the bgw protocol for perfectly-secure multiparty computation. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, page 36, 2011.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in CryptologyCRYPTO'98*, pages 26–45. Springer, 1998.
3. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in CryptologyASIACRYPT 2001*, pages 566–582. Springer, 2001.
4. Mihir Bellare and Phillip Rogaway. Asymmetric Encryption. `http://cseweb.ucsd.edu/~mihir/cse207/w-asym.pdf`.
5. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.
6. J.M. Bohli and A. Pashalidis. Relations among privacy notions. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):4, 2011.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.
8. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1), 1988.
9. D.L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
10. Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.
11. S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. 2003.
12. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

13. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.
14. A. Hevia and D. Micciancio. An indistinguishability-based characterization of anonymous channels. In *Privacy Enhancing Technologies*, pages 24–43. Springer, 2008.
15. A. Pashalidis. Measuring the effectiveness and the fairness of relation hiding systems. In *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, pages 1387–1394. IEEE, 2008.
16. A. Pfitzmann and M. Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. *Website, February*, 2008.
17. A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *URL: http://dud. inf. tu-dresden. de/literatur/Anon_Terminology_v0*, 34, 2010.
18. A. Pfitzmann and M. Köhntopp (Hansen). Anonymity, unobservability, and pseudonymitya proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.
19. M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, 16(4):482–494, 1998.
20. Jian Ren and Jie Wu. Survey on anonymous communications in computer networks. *Computer Communications*, 33(4):420–431, 2010.
21. K. Sampigethaya and R. Poovendran. A Survey on Mix Networks and Their Secure Applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006.
22. A. Serjantov, R. Dingledine, and P. Syverson. From a trickle to a flood: Active attacks on several mix types. In *Information Hiding*, pages 36–52. Springer, 2003.
23. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
24. I. Vajda. Uc framework for anonymous communication. Technical report, Cryptology ePrint Archive Report 2011, 2011.
25. M. Veeningen, B. De Weger, and N. Zannone. Modeling identity-related properties and their privacy strength. *Formal Aspects of Security and Trust*, pages 126–140, 2011.
26. Matthew K Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Transactions on Information and System Security (TISSEC)*, 7(4):489–522, 2004.

A Theorem 3's Proof

Proof. (sketch) Let π be some deterministic protocol that ensures Comp-**SA**-anonymity against malicious destination that is also global eavesdropper. If some participant p_i of π, sends traffic according to the number of messages in its application queue, a global eavesdropper attacker can detect that, by choosing two different scenarios where p_i sends different amount of messages in its application queue.

Therefore, π's participants send regardless the messages in their application queue, and for each participant p_i, for every scenario σ and every R, $L_i(\sigma, R)$ is some constant.

Assume on the contrary that there are some σ' and R' such that $MaxOut_{\sigma',R'} \geq min\{L_i(\sigma', R')|p_i \text{ is a honest potential sender}\}$. Let $L_i(\sigma', R')$ get minimal value when $i = j$; namely, p_j is the participant with the lowest $L_i(\sigma', R')$ value.

Let σ' be described by $\{M_i^{(0)}\}_{i=1}^{R'}$.

For every $M_i^{(0)}$ matrix we define $M_i^{(1)}$ as follows: for every $i \in [R']$, $l \in [n]$, $M_{i\,j,l}^{(1)} = \cup_{k=1}^{n} M_{i\,k,l}^{(0)}$, i.e., p_j sends all the messages that were sent by $M_i^{(0)}$ to the same destinations. Obviously, for every $i \in [R']$, $(M_i^{(0)}, M_i^{(1)}) \in \widehat{R}_{\mathbf{SA}}^{H}$.

We now consider the following run of the Comp-**SA**-b experiment (Alg 1): The attacker simulates the experiment to R' rounds such that every round it chooses $(M_i^{(0)}, M_i^{(1)})$. During the simulation, it acts as a honest participant, but count the messages that reach the malicious destination in some counter C. In the end of the R' rounds, if $C > L_j(\sigma', R')$ the attacker returns 0, and otherwise returns 1.

Because π is deterministic, if $b = 0$ then from the choice of σ' and R', $C > MaxOut_{\sigma',R'} = L_j(\sigma', R')$, and if $b = 1$ then $C \leq L_j(\sigma', R')$, as p_j sent all the messages. Therefore the attacker has the maximal advantage (Definition 2), 1, and π is not Comp-**SA**-anonymous against malicious destination that is also global eavesdropper. In contradiction to the initial assumption.

This proves the first claim of the theorem. The second claim, derived directly from the definition of $L_i(\sigma, R)$ and from the first claim. □

B DC-Net's CompSA-Anonymity Against Malicious Destination

B.1 DC-Net

The dining-cryptographers network protocol [8], is a multi party computation protocol. The protocol is based on Chaum's solution to the dining cryptographer problem: Three cryptographers gather around a table for dinner. The waiter informs them that the meal has been paid by someone, who could be one of the cryptographers or the National Security Agency (NSA). The cryptographers respect each other's right to make an anonymous payment, but want to find out whether the NSA paid. In the solution, every cryptographers flips a coin (or a bit) and shows his result (1 or 0) only to the cryptographer on his left. Now every cryptographers should publish the XOR of his own bit with the bit of the cryptographer on his right side. The cryptographer who paid for the meal (if any) should XOR his answer with 1. Now, simply, if the XOR between all the published bits is 0 then NSA paid for the meal, otherwise, it is one of the cryptographers.

To send messages of length l, a random bits vector of length l should be chosen. The protocol can be extended to n peers in different topologies, the most common is the ring.

B.2 DC-Net is Not Comp-SA-Anonymous Against Malicious Destination

There is something that the DC-net protocol cannot hide: whether in a round two participant sent or only one. In the DC-net, it takes one round to send a message, and only one participant can send a message in a round (otherwise, there is a collision).

We now consider the following scheme: the three cryptographer p_1, p_2, p_3 want to send anonymous messages to a fourth cryptographer p_4 ($n = 4$). For that purpose, they run the DC-net protocol in rounds between them, and every one of them sends his output to the destination. The destination XORs the three cryptographers output and gets the message.

We present a malicious destination attacker that has non-negligible advantage. The attacker works as follows:

1. In the first round, choose two matrices: in the first scenario p_1 and p_2 send m_1 and m_2 (such that $m_1 \oplus m_2 \notin \{m_1, m_2\}$) respectively, and in the second scenario p_1 sends both the messages (these matrices are legal by $\widehat{R}^H_{\mathbf{SA}}$).
2. After the three cryptographers send their first outputs c_1, c_2, c_3, calculate $m' = c_1 \oplus c_2 \oplus c_3$. If $m' \in \{m_1, m_2\}$ return 1. Otherwise return 0.

$\mathcal{A}$ is a polynomial time. And additionally: $Adv^{Comp-\mathbf{SA}}_{DC-net,4,\mathcal{A},(\{4\},\emptyset,1)}(k) = |Pr[Expt^{Comp-\mathbf{SA}-1}_{DC-net,4,\mathcal{A},(\{4\},\emptyset,1)}(k) = 1]\text{-}Pr[Expt^{Comp-\mathbf{SA}-0}_{DC-net,4,\mathcal{A},(\{4\},\emptyset,1)}(k) = 1]| = 1$

Therefore according to the definition of Section 3 DC-net is not Comp-**SA**-anonymous. We note that while the destination does not eavesdrop and does not control some of the peers, collision detection mechanism might be useful. However, such mechanisms might hurt the unobservability of the protocol against malicious peers.

B.3 DC-Net is Comp-$\widehat{\mathbf{SA}}$-Anonymous Against Malicious Destination

We now discuss a scheme of $n > 4$ participants ($n - 1$ potential senders and destination p_n). We give a proof sketch that in a ring topology, while the channels are pairwise encrypted with secure encryption scheme [4], DC-net is Comp-$\widehat{\mathbf{SA}}$-Anonymous by the malicious destination extension of Section 6, even against malicious destination and global eavesdropper attacker that controls additional peer. Namely, an attacker with capability Cap=$(\{i, n\}, [n], 1)$ for some $i \in [n-1]$.

We omit here the proof for the following claim: given the messages that were sent in a round, and given the final output of all the participants, it is impossible to learn something about the senders identity (unconditional anonymity). This claim holds even if one participant is malicious: i.e., tell the malicious destination what he sent and received [11, Appendix A]. In a ring topology (of more than three peers), for breaking some peer's anonymity, there is a need in both the peers on its sides [26].

Following the above claim, it is enough to prove that if the attacker cannot break the encryption scheme, for every two scenarios with the same unprotected data, in every round in both the scenarios the same messages are sent (scenarios with the same unprotected data are defined by two matrices sequences $\{M_i^{(0)}\}_{i=1}^{s}$ and $\{M_i^{(1)}\}_{i=1}^{s}$, such that for every $1 \leq i \leq s$, $(M_i^{(0)}, M_i^{(1)}) \in R_{\mathbf{SA}}^{H,\tau}$ for some permutation τ over $|H|$ elements).

But by the $R_{\mathbf{SA}}^{H,\tau}$ relation, in every round, when p_j sends m in the first scenario, then $p_{\tau(j)}$ sends m in the second scenario, and therefore the messages that are sent are identical for every round in both the scenarios.

Therefore, the attacker cannot break the anonymity without breaking the encryption scheme for learning additional information. Formal proof could be done by reducing the Comp-$\widehat{\mathbf{SA}}$-anonymity to the security of the encryption scheme.

C Controlling the Application Adaptively Matter

We now present a toy example of a protocol that ensures unobservability by [14] due to the inability of the adversary in their experiment to manipulate the application level adaptively, but not according to our definition:

The toy protocol for sending one message, is only for two participants, Alice and Bob. The protocol's run takes four iterations:

1. Alice sends a constant length message, m, encrypted by a CPA-Secure encryption scheme [4]. The message comes from the application, or is a dummy if the application has nothing to send.
2. Bob sends Alice a plain text with a random identifier $\in \{0,1\}^k$
3. Alice can send another message to Bob.
4. If Bob got the random identifier from Alice in the third iteration, he sends her m as a plain-text.

A passive global eavesdropper attacker that have non-negligible advantage according to definition (3), just inserts different messages into Alice's application queue in each scenario. After inspecting the random identifier that Bob sent in the second iteration, he adds the message identifier to Alice's application queue in both the scenarios, and guesses (with success probability 1) the simulated scenario by comparing Bob's response to the first message that Alice sent in each scenario.

But, according to [14]'s definition, the above toy protocol achieves unobservability. We shortly bring a proof sketch: For knowing the which scenario was simulated, the attacker has two options: Either to break the IND-CPA security of the encryption scheme, or to guess the random identifier (it has only one guess) with negligible success probability. Therefore, a simple reduction from the CPA-security of the protocol's encryption scheme to the **UO**-anonymity of the toy protocol will do the work.

D Example: Tor Is Not Comp-N-Anonymous Against Local Eavesdropper For Any Anonymity Notion N

In this appendix, we bring an example to anonymity protocol that its anonymity notions relies mainly on a lot of traffic, and show that it is not Comp-**N**-anonymous for any anonymity notion **N** (for some anonymity notion **N** [14]). We examine a simplified version of the Tor protocol [10], denoted by *Simp-Tor*.

In Simp-Tor there are three types of participants: clients, routers, and external sites. Every N iterations, every client chooses uniformly three different routers. A client who wants to send messages to sites (that are not necessary Tor clients), use the chosen three routers as a path, such that the last router sends the encrypted (even with perfect encryption scheme) messages to their destinations. The protocol is described in Algorithms 3 and 4.

Algorithm 3 *IterationAction*(iteration t). Client's iteration action in Simp-Tor.

if $t \bmod N = 0$ **then**
  $c_1, c_2, c_3 \leftarrow$ Choose 3 random routers
end if
$\{dest_i, m_i\}_{i=1}^{l} \leftarrow$ All the application messages.
Send to c_1: $\mathcal{E}_{PK_{c_1}}(c_2, \mathcal{E}_{PK_{c_2}}(c_3, \mathcal{E}_{PK_{c_3}}(\{dest_i, \mathcal{E}_{PK_{dest_i}}(m_i)\}_{i=1}^{l})))$

Algorithm 4 Router c iteration action in Simp-Tor

for each $\mathcal{E}_{PK_c}(NextDest, m)$ **do**
  Send m to $NextDest$.
end for

For testing the protocol against local eavesdropper to one of the routers, we choose the next parameters for the experiment: π is the protocol. π's setup procedure returns a sequence of n states such that the first two states are of two Tor clients, the last two states are of two external sites (only receive messages and do not participate in the protocol), and the rest of the states are of Tor routers. π's setup initialize the states such that all the protocol participants know the relevant other identities and cryptographic keys. We only limit the n parameter such that $n \geq 7$ (there are at least 3 Tor routers). $\mathcal{A}$ is the $\mathcal{PPT}$ attacker, and its capability is $Cap = (\emptyset, \{3\})$. I.e., the attacker eavesdrops only to one participant - the one with $STATE_3$, which is one of the Tor routers.

We now present an attacker, $\mathcal{A}$, that for any **N**-anonymity notion [14] have non-negligible advantage according to definition (3); i.e., for every negligible function $negl$:

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

$\mathcal{A}$ works as follows:

1. Choose a random message $m' \in V$. Choose $M^{(0)}$ to be empty, except $m^{(0)}_{1,n-1} = m^{(0)}_{2,n} = \{m', m'\}$, and $M^{(1)}$ to be also empty, except $m^{(1)}_{1,n-1} = m^{(1)}_{1,n} = m^{(1)}_{2,n-1} = m^{(1)}_{2,n} = \{m'\}$.
2. In all the other iterations, choose $M^{(0)}$ and $M^{(1)}$ to be empty.
3. After the end of the fourth iteration, when the messages already reached their destinations, if the eavesdropped router sent 2 messages and to the same site, return 0. if it sent only 2 messages, but to different sites, return 1. Otherwise return random bit.

We now prove that the above $\mathcal{A}$ breaks the definition for every $\mathbf{N}$-anonymity. First of all, all the pairs of matrices that were chosen by the attacker are in the $R^H = R^H_{f_\cup} \cap R^H_{f^T_\cup}$ relation, that contains $R^H_{\mathbf{N}}$ for each $\mathbf{N}$.

We denote the number of Tor routers in the setup by c. $c = n-4$ is polynomial in the security parameter k. $\mathcal{A}$ wins for sure, only if exactly one Tor client chose the eavesdropped router to be the last in his path. As the routers in the paths are chosen uniformly, the probability for such an event is $2 \cdot (\frac{1}{c} \cdot \frac{c-1}{c})$ for $c \geq 3$. In all the other cases $\mathcal{A}$ guesses, so its probability to win is $\frac{1}{2}$. therefore:

$$Pr[Expt^{Comp-\mathbf{N}-0}_{Simp-Tor,n,\mathcal{A},Cap}(k) = 1] = \frac{1}{2} \cdot (1 - 2 \cdot \frac{c-1}{c^2})$$

$$Pr[Expt^{Comp-\mathbf{N}-1}_{Simp-Tor,n,\mathcal{A},Cap}(k) = 1] = \frac{1}{2} \cdot (1 - 2 \cdot \frac{c-1}{c^2}) + 2 \cdot \frac{c-1}{c^2})$$

Therefore, for every negligible function, $negl$, the advantage of $\mathcal{A}$ is:

$$Adv^{Comp-\mathbf{N}}_{Simp-Tor,n,\mathcal{A},Cap}(k) = 2 \cdot \frac{c-1}{c^2}) \geq negl(k)$$

The last inequality is because $\frac{c-1}{c^2} \in poly(k)^{-1}$ which is not negligible in k. The same attack could be done when the attacker controls one of the routers. In similar algorithm that chooses routers path per destination, a similar attack will work; in this attack, the attacker wins if the eavesdropped or compromised router is chosen to be the first in one of the paths.

E Efficient Relaxed Ultimate Anonymity and t-Liveness

We present the Disjoint Paths Protocol (DPP), that ensures relaxed anonymity against strong malicious destination that controls $t < \sqrt{n}$ of the participants. DPP also satisfies t-liveness for $t < \sqrt{n}$.

For simplicity, we discuss a scheme where n participants need to send anonymous messages to one (malicious) destination, d.

The main problem of mixnet based protocol is malicious peers who drops messages of specific honest peers; then the destination can distinguish between the scenarios by the messages it receives. Duplication of messages or using secret sharing [23] might not help, as the destination or the peers it controls, can still

distinguish between the scenarios by the number of duplicated messages or the number of message's shares that reach the destination.

DPP overcome dropping or changing of messages by malicious participants by using $t+1$ disjoint paths to every relay. To overcome t malicious peers, DPP's routing uses $(t+1)^2$ peers as relays. The relays are sorted into $t+1$ levels of $t+1$ relays each. We denote the levels by l_i, and the j-th relay in l_i by $r_{i,j}$.

We now describe DPP's routing in high level: every round, all the potential senders send one message (real or dummy) to every relay in the first level (l_1). To deal with messages dropping or changing, in every l_i, $1 \leq i \leq t+1$ every relay collects all the messages it received (one copy for each message), and sends them randomly permuted, to every relay in l_{i+1}. The relays in l_{t+1} send the messages to the destination. Figure 3 depicts the routing procedure.

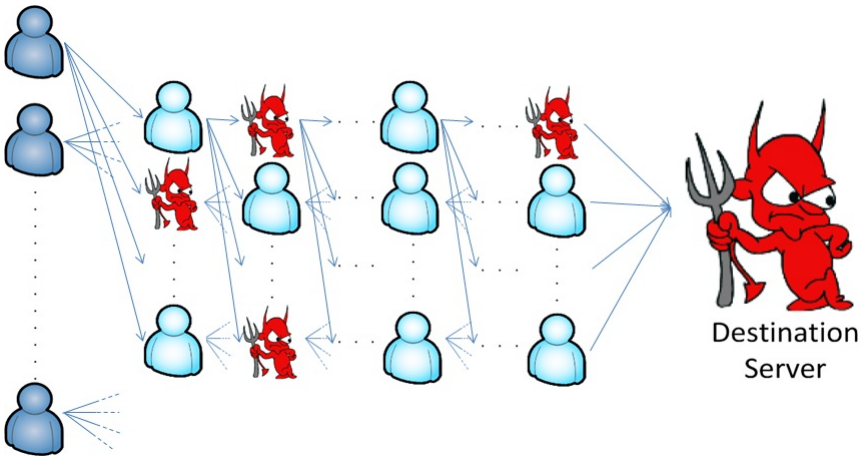

Fig. 3: DPP's routing.

To prevent learning of the messages content (dummy or real, and any other information) and to enforce the routing, and prevent a case where malicious peer directly sends the message to the destination, the messages are encrypted with semantically secure encryption scheme [4], such that like in the Onion Routing protocol [19], the path of a message must be followed in order to get the content.

Unlike onion-routing, DPP has different $(t+1)^{t+1}$ paths for each message. Consequently, creating onions with each relay's public key is impossible. Therefore one common public key (and appropriate private key) is shared among the relays in the same level: (pk_i, sk_i) are the keys of the relays in l_i, and the onion of each message is done by encrypting the message with the destination's public key, and then with all pk_is from $i = t+1$ to 1.

The protocol can work also with symmetric encryption scheme (more efficient), assuming the senders knows all the keys, and every relay knows only its level key.

Every layer contains also the round where the message should be sent. That is to prevent reply attack: malicious relays in l_1 can send twice (in two different iterations) the same onion it got from some peer, and when the destination gets two identical messages, the originator will be exposed.

Algorithm 5 DPP's senders pseudocode. In the tuple (C, R, D), C is the message (possible ciphertext) to forward, R is the round to forward the C, and D is the next destination (the level of the next relays or the final destination).

```
    // round is the current round
 1: (m, d) ← Application.GetMessage()
 2: if m = null then
 3:     m ← dummy
 4: end if
 5: (C, R, D) ← (E_{pk_d}(m), round + t + 1, d)
 6: for i = t + 1 to 1 do
 7:     (C, R, D) ← (E_{pk_i}(C, R, D), r + i − 1, i)
 8: end for
 9: for j = 1 to t + 1 do
10:     send C to r_{1,j}
11: end for
```

E.1 DPP Satisfies Relaxed Ultimate Anonymity and *t*-Liveness

We examine the relays as a $t+1 \times t+1$ matrix M, such that $M_{i,j} = r_{i,j}$. We denote an encrypted onion by O_0. We denote the partially peeled onion, after the relays in l_i decrypted their layer by O_i.

***t*-liveness** Every honest relay receives all the real messages (as onions). That is, because every M's row contains $t+1$ relays, and at most t of them are malicious, so there is a honest relay in every row. The honest relays in l_1 send all the real messages to all the relays in l_2, and from that point, every honest relay in l_i sends all the messages he received (including all the real messages, and maybe with additional messages added by malicious relays) to the relays in l_{i+1}. Therefore, in l_{t+1} there is at least one honest relay that receives all the messages and forward them to their destinations.

Comp-$\widehat{\text{SA}}$-Anonymity Due to the $R^{H,\tau}_{\widehat{\mathbf{SA}}}$ relation, in every round the same messages are sent, such that the only difference is in the senders. Therefore for breaking the anonymity, an attacker must connect or disprove connection between a message and its sender.

We relies on the following claim: Given an CPA secure encrypted message, it is computationally hard to learn about the message content [4].

Algorithm 6 DPP's pseudocode for the relay $r_{i,j}$.

Input: $t+1$ vectors of l shuffled encrypted messages. The relays in the first level (l_1) receives only l messages (referred as one vector).

Output: Vector of all the input messages without the outer encryption layer, to the next hop of the messages. The relays in l_{t+1} send the messages to their destinations.

```
// round is the current round
Set.init() // Initialize empty set of messages
for each messages-vector v in the input do
    for k = 1 to |v| do
        (C, R, D) ← D_{sk_i}(v[k])
        if round = R and (D = i + 1 or i = t + 1) then
            Set.add(C, D)
        end if
    end for
end for
if i < t + 1 then
    Vector.init(Set) // Create messages-vector from Set
    Vector.shuffle() // Shuffle the messages in Vector
    for k = 1 to t + 1 do
        send Vector to r_{i+1,k}
    end for
else
    for each message (C, D) in Set do
        Send C to D
    end for
end if
```

Consequently, given two randomly shuffled sets of onions: $S_i = \{O_i^1, O_i^2, ..., O_i^p\}$ and $S_{i+1}\{O_{i+1}^1, O_{i+1}^2, ..., O_{i+1}^p\}$, such that the encryptions of the layers were done with the same keys for all the onions (onions of honest senders), the probability of an attacker to connect between some O_i^j to O_{i+1}^{j+1} is negligible. Otherwise, by a simple reduction, the encryption scheme is not CPA secure.

Because the number of malicious relays is less then $t+1$, in M there is at least one level without malicious relays. We denote the lowest index of such a level with h. From the routing procedure (see also the t-liveness proof), when the relays in l_h receives the messages, every one of them has all the real messages. From the security of the encryption scheme, from that point the attacker cannot link the onions. Every relay in l_h sends all the real messages shuffled (in one copy) to l_{h+1} such that from that point, the originators of the messages cannot be linked to the messages. Similarly to cascade mixnet [9].

Comp-UO-Anonymity As described above, DPP satisfies unobservability only if there is only one destination. Otherwise, an attacker can distinguish between two scenarios by destinations of the messages. To deal with many destinations, there is a need to remove the final destination from the message, and that the relays in l_{t+1} will send all the messages to all the possible destinations. In such a case, the encryption scheme should be also IK-CPA [3]. Namely, the attacker cannot distinguish between two messages that were encrypted by different keys. The unobservability relies only on the constant sending rate and the IND-CPA and the IK-CPA security of the encryption scheme. We omit here the reduction from the Comp-**UO**-anonymity to the CPA-security of the encryption scheme.

E.2 Communication Complexity

Every message is encrypted to an onion. We analyze the communication complexity in the number of times an onion is sent. Every onion is sent once to each of the $t+1$ relays in l_1, and $t+1$ times to each relay in the other levels. The total number of 'send' events is therefore $t+1+t \cdot t \cdot (t+1) \in O(t^3)$. For every $t < \sqrt{n}$, the communication overhead is therefore $O(n^{1.5})$.