Journal of Privacy and Confidentiality (2016) **7**, Number 2, 79–125

AnoA: A Framework for Analyzing Anonymous Communication Protocols

Michael Backes,* Aniket Kate,† Praveen Manoharan,‡ Sebastian Meiser §
and Esfandiar Mohammadi ¶

Abstract. Anonymous communication (AC) protocols such as the widely used Tor network have been designed to provide anonymity over the Internet to their participating users. While AC protocols have been the subject of several security and anonymity analyses in the last years, there still does not exist a framework for analyzing these complex systems and their different anonymity properties in a unified manner.

In this work we present AnoA: a generic framework for defining, analyzing, and quantifying anonymity properties for AC protocols. In addition to quantifying the (additive) advantage of an adversary in an indistinguishability-based definition, AnoA uses a multiplicative factor, inspired from differential privacy. AnoA enables a unified quantitative analysis of well-established anonymity properties, such as sender anonymity, sender unlinkability, and relationship anonymity. AnoA modularly specifies adversarial capabilities by a simple wrapper-construction, called adversary classes. We examine the structure of these adversary classes and identify conditions under which it suffices to establish anonymity guarantees for single messages in order to derive guarantees for arbitrarily many messages. This then leads us to the definition of *Plug'n'Play adversary classes* (PAC), which are easy-to-use, expressive, and satisfy this condition. We prove that our framework is compatible with the universal composability (UC) framework and show how to apply AnoA to a simplified version of Tor against passive adversaries, leveraging a recent realization proof in the UC framework.

Keywords. Anonymous Communication, Anonymity Metric, Relationship Anonymity, Unlinkability

1 Introduction

Protecting individuals' privacy in online communications has become a challenge of paramount importance. A wide variety of privacy enhancing technologies, comprising many different approaches, have been proposed to solve this problem. Among these, anonymous communication (AC) protocols seek to protect users' privacy by anonymizing their communication over the Internet. Employing AC protocols has become increasingly popular over the last decade. This popularity is exemplified by the success of the Tor network (Tor Project).

*CISPA, Saarland University, Germany. mailto:backes@cs.uni-saarland.de
†Purdue University, USA mailto:aniket@purdue.edu
‡CISPA, Saarland University, Germany. mailto:manoharan@cs.uni-saarland.de
§CISPA, Saarland University, Germany. mailto:meiser@cs.uni-saarland.de
¶CISPA, Saarland University, Germany. mailto:mohammadi@cs.uni-saarland.de

It is, however, often unclear how well such AC protocols actually protect the anonymity of an individual's network communication. This is due to AC protocols often being complex decentralized cryptographic algorithms that require careful analyses for at least partial security guarantees under restricted settings: there has been significant previous work on analyzing the anonymity provided by various AC protocols, such as dining cryptographers networks (DC-nets) (Chaum, 1988), Crowds (Reiter and Rubin, 1998), mix networks (Mixnets) (Chaum, 1981), and onion routing (e.g., Tor) (Reed et al., 1998). However, most of the previous work only consider a single anonymity property for a particular AC protocol under a specific adversary scenario (Gierlichs et al., 2008; Feigenbaum et al., 2007b; 2012), and previous frameworks (Syverson et al., 2000; Díaz et al., 2002; Serjantov and Danezis, 2002; Shmatikov, 2004; Mauw et al., 2004; Halpern and O'Neill, 2005; Shmatikov and Wang, 2006; Díaz, 2006; Hughes and Shmatikov, 2004) do not consider the computational aspects of the cryptographic realization of AC.

We observe that a general approach to quantifying the anonymity provided by AC protocols can pave the way for a unified analysis of various AC protocols, identifying common and unique weaknesses of existing approaches, and help in developing novel approaches to anonymous communication that circumvent these problems. Such a general approach, however, is to provide means to analyze AC protocols under different anonymity notions and various degrees of adversarial capabilities, as well as taking into account computational risks presented by the deployed cryptography.

As the main contribution of this work, we present a novel anonymity analysis framework AnoA. AnoA allows us to define anonymity properties in a unified and comparable manner, with an anonymity definition based on indistinguishability of probabilistic processes, inspired by similar notions utilized in different contexts, such as differential indistinguishability (Backes et al., 2015a) for cryptography with imperfect randomness, differential privacy (Dwork, 2006; Dwork et al., 2006) or for privacy in statistical databases. We adopt several strengths of these earlier differential indistinguishability notions that are suited for reasoning about anonymity: (a) a strong adversary that has maximal control over two different (adjacent) settings that it has to distinguish and (b) an inherent quantitative aspect that does not only allow us to qualitatively, but also quantitatively assess the anonymity provided by AC protocols, by bounding distinguishability of probabilistic processes through additive, as well as multiplicative factors.

The AnoA framework provides anonymity guarantees for various anonymity properties such as sender anonymity, recipient anonymity, or sender unlinkability through explicit *anonymity functions* α that formalize each of these anonymity notions in the AnoA framework: guarantees determined through AnoA are then relative to the specific anonymity function deployed. In contrast to previous work on anonymity properties, AnoA formulates anonymity properties in a game-like notion, in which the adversary can choose all messages sent through the network except a anonymity challenge message—which results in a strong adversary.

Moreover, AnoA is compatible with simulation-based composability frameworks, such as UC (Canetti, 2013), GNUC (Hofheinz and Shoup, 2013), IITM (Küsters and

Tuengerthal, 2013), RSIM (Backes et al., 2007), or even time-sensitive variants, such as TUC (Backes et al., 2014b). In particular, for all protocols that are securely abstracted by an ideal functionality (Wikström, 2004; Camenisch and Lysyanskaya, 2005; Danezis and Goldberg, 2009; Kate and Goldberg, 2010; Backes et al., 2012), our definitions allow an analysis of these protocols in a purely information theoretical manner.

As a second contribution, we formalize the well-established notions of sender anonymity, (sender) unlinkability, recipient anonymity and relationship anonymity in our framework, by introducing appropriate anonymity functions. We discuss why our anonymity definitions accurately capture these notions, and show for sender anonymity, (sender) unlinkability and recipient anonymity that our definition is equivalent to the definitions from the literature. For relationship anonymity, we argue that previous formalizations captured recipient anonymity rather than relationship anonymity, and we discuss the accuracy of our formalization.

As a third contribution, we show how we can modify adversarial capabilities in our anonymity game by a simple extension of AnoA with adversary classes: these adversary classes function as a wrapper around the actual adversary, restricting the information the adversary gains from the AC protocol, as well as the set of actions the adversary is able to perform. These adversary classes allow us to represent more realistic and practical adversaries and provide anonymity guarantees relative to these adversaries' capabilities. We examine the structure of these adversary classes and provide insight into the necessary conditions to allow these adversary classes to be composable. This then leads us to the definition of *Plug'n'Play adversary classes* (PAC), which are easy-to-use and allow for composition.

Finally, as the fourth contribution, we apply our framework to the practically relevant AC protocol Tor. We leverage previous results that securely abstract Tor as an ideal functionality (in the UC framework) (Backes et al., 2012). Then, we illustrate that proving sender anonymity, sender unlinkability, and relationship anonymity against passive adversaries boils down to a combinatorial analysis, purely based on the number of corrupted nodes in the network. We further leverage our framework and analyze the effect of a known countermeasure for Tor's high sensitivity to compromised nodes: the entry guards mechanism.We discuss that, depending on the scenario, using entry guards can have positive or negative effect. This basic analysis has been used as a stepping-stone for successful follow-up work, such as the real-time anonymity monitor proposed in Backes et al. (2014a) and for analyzing the impact of different path selection algorithms and different types of adversaries on Tor anonymity (Backes et al., 2015b).

Outline. We begin by discussing the related work in Section 2. After establishing the basic notation used throughout the paper in Section 3, we introduce the AnoA framework in Section 4 where we describe its basic structure and the main components. In Section 5, we then provide formalizations of various anonymity notions in AnoA before we discuss adversary classes and their properties in Section 6. Finally, we utilize the AnoA formalization of anonymity to provide anonymity guarantees for the Onion Routing (OR) protocol based on an analysis of an ideal functionality of the OR protocol

in Section 7. In Section 8, we conclude the paper and give an outlook on possible future directions. In the appendix, we provide full proofs of theorems stated in this paper, and show that UC-realization preserves anonymity as defined in AnoA.

2 Related and Prior Work

Pfitzmann and Hansen (2010) develop a consistent terminology for various relevant anonymity notions; however, their definitions lack formalism. Nevertheless, these informal definitions form the basis of almost all recent anonymity analysis, and we also adopt their terminology to validate the anonymity definitions in AnoA.

There have been analyses which focus on a particular AC protocol, such as Serjantov and Danezis (2002); Díaz (2006); Shmatikov and Wang (2006); Gierlichs et al. (2008) for Mixnet, Bhargava and Palamidessi (2005); Andrés et al. (2011) for DC-net, Díaz et al. (2002); Shmatikov (2004) for Crowds, and Syverson et al. (2000); Mauw et al. (2004); Feigenbaum et al. (2007a;b; 2012) for onion routing. Most of these study a particular anonymity property in a particular scenario and are not flexible enough to cover the emerging system-level attacks on the various AC protocols. Some of the existing frameworks and analyses focus on changes in entropy Díaz et al. (2002); Serjantov and Danezis (2002); Shmatikov and Wang (2006); Díaz (2006), which allows, but also requires, to explicitly specify the adversary's initial knowledge. (We refer the readers to Feigenbaum et al. (2012, Sec. 5) for a detailed survey.)

Notably, there has been work on exploring metrics related to the multiplicative factor in our anonymity definitions: Andrés et al. (2011) distinguish multiplicative leakage and additive leakage (however, as alternatives, not in combination with each other). Their approach is focused on concurrent systems and applied to the dining cryptographers protocol. Moreover, Shmatikov Shmatikov (2004) defines the notions of "probable innocence" and "possible innocence" – in terms of our metric, a multiplicative factor quantifies the strength of "probable innocence", while ensuring "possible innocence", whereas the distinguishing events rule out "possible innocence".

The most recent result (Feigenbaum et al., 2012) among these by Feigenbaum, Johnson and Syverson models the OR protocol in a simplified black-box abstraction, and studies a notion of relationship anonymity which is slightly different from ours: here the adversary wishes to identify the destination of a user's message. As discussed in Section 5.2, this relationship anonymity notion is slightly weaker than ours. Moreover, their model is not flexible enough to extend to other system-level scenarios such as fingerprinting attacks (Panchenko et al., 2011; Cai et al., 2012; Dyer et al., 2012).

Building on the work of Hevia and Micciancio (2008), Gelernter and Herzberg published, concurrently with this work, an expressive framework that extends the work of Hevia and Micciancio with active adversaries that adaptively send inputs (Gelernter and Herzberg, 2013). They apply their methodology to obtain an impossibility result for a strong combination of sender anonymity and unobservability, which they coin "ultimate anonymity". However, as in the work of Hevia and Micciancio, they only consider qualitative anonymity notions (i.e., protocols that only allow a negligible chance

of de-anonymizing users), which does not allow them to quantify the anonymity loss in low-latency anonymous communication networks, such as Tor. Moreover, they do not provide a mechanism, such as the adversary classes presented in this work, to flexibly relax the strength of the adversary.

In 2013, the first version of AnoA (Backes et al., 2013) introduced the concept of anonymity in terms of indistinguishability with a multiplicative factor. However, the formal description was closer to differential privacy: it described AC protocols as mechanisms running on databases of user actions. Thus, this version did not capture interactive scenarios in a meaningful manner. Moreover, the adversary was fixed to a Turing machine that sent two databases and then tried to distinguish between them. For their variant of AnoA they showed single-challenge reducibility without the concept of adversary classes. From this work we have only used the intuitive idea of a multiplicative quantification for AC protocols, the UC realization proof for `IND-ANO`, and the example application of AnoA to an idealized version of Tor.

In a follow-up in 2014, AnoA was extended to adaptive adversaries (Backes et al., 2014a). They also introduced the concept of *adversary classes* as a means to adapt the strength of the adversary to more realistic (real-world) scenarios, and analyzed which properties an adversary class requires for the game to still provide single-challenge reducibility. However, the main contribution of Backes et al. (2014a) was an application of AnoA to a realistic instance of Tor, based on Tor's real path selection algorithm and its network status. From this work we adopt the adaptive description of AnoA, as well as the underlying concept of (single-challenge reducible) adversary classes.

Recently, in an even more in-depth analysis, AnoA was again applied to Tor (Backes et al., 2015b) for analyzing different variants of Tor's path selection algorithm against a variety of different structural adversaries. All these adversaries were described in terms of (single-challenge reducible) adversary classes, which they derived from a relatively general *budget-adversary* (see Example 7). From this work we only used the idea of deriving (even more) general adversary classes, which can easily be instantiated.

3 Notation

Before we present AnoA, we briefly introduce some of the notation used throughout the paper. We differentiate between two different kinds of assignments: $a := b$ denotes a being assigned the value b, and $a \leftarrow \beta$ denotes that a value is drawn from the distribution β and a is assigned the outcome. In a similar fashion $i \xleftarrow{\mathcal{U}} I$ denotes that i is drawn uniformly at random from the set I.

Probabilities are given over a probability space which is explicitly stated unless it is clear from context. For example $\Pr[b = 1 : b \xleftarrow{\mathcal{U}} \{0,1\}]$ denotes the probability of the event $b = 1$ in the probability space where b is chosen uniformly at random from the set $\{0,1\}$.

Our security notion is based on interacting Turing Machines (TM). We denote with $\langle A|B\rangle$ the interaction of two TMs A and B: the interaction starts with the activation

of A and both machines activate each other by putting an input message in the other machine's input tape. The interaction ends when A produces an output.

In this paper we focus on computational security, i.e. all machines are computationally bounded. More formally, we consider *probabilistic, polynomial time* TMs, which we denote with `ppt` whenever required.

4 The AnoA Framework

AnoA quantifies how imperfect an anonymous communication protocol is compared to a perfect anonymous channel that delivers all messages without leaking any information about the communication. AnoA solely quantifies communication anonymity and does not address privacy-loss via semantic means, e.g., if users sends its IP address to a malicious server.

In the AnoA framework, we formalize anonymity through a challenge-response game, where a *challenger* machine `Ch` (which internally runs a protocol Π) interacts with a ppt *adversary* machine $\mathcal{A}$. We assume a strong adversary that knows the behavior of all protocol parties except for some specific challenges for which the adversary has to guess correctly. As in other cryptographic definitions, we over-approximate the adversary's knowledge by assuming that the adversary $\mathcal{A}$ determines the inputs for the protocol parties by sending instructions (i.e., which party S sends a message m to which recipient R). The challenges are chosen by the adversary and constitute a pair of such instructions, and the challenger chooses one of them, depending on a secret input of the challenger (the challenge bit b). Naturally, the adversary, in addition, learns information from observations made during execution of the protocol. These observations can include intercepted messages through compromised parties in the network, traffic observations on links between parties, and other types of leakage, and are formalized in Π. The adversary wins the game if it is able to guess the challenge bit correctly.

For capturing different anonymity notions, the challenger internally processes challenge messages (red box inside `Ch`). How challenge messages are processed in detail depends on the anonymity notion currently considered: e.g. for sender anonymity, a specified message is sent to a specified recipient by one of two specified senders. The anonymity provided by Π can now be measured by the advantage $\mathcal{A}$ has in guessing which alternative `Ch` chose by observing the output of Π.

We over-approximate the capabilities of the adversary, which sometimes leads to adversaries that are too strong. In order to capture more realistic or more specific scenarios with weaker adversaries, we introduce restrictions, i.e., a machine that internally runs another arbitrary machine (i.e., the adversary). We call this outer, restricting machine C an *adversary class*.

The adversary $\mathcal{A}$ inside an adversary class C, the challenger `Ch`, and the protocol Π are represented by interacting probabilistic, polynomial time (`ppt`) Turing machines. The overall structure of the protocol simulation in AnoA is illustrated in Figure 1.

The AnoA framework is designed to also yield useful guarantees for imperfect AC protocols (such as Tor) by quantifying the imperfectness of an AC protocol. As our definition comprises a (in)distinguishability game, we could, in principle, simply consider the additive distance between the advantage of an adversary, as done in other computational indistinguishability definitions. In order to achieve a qualitatively more accurate characterization, however, we also include a multiplicative factor inspired by differential privacy. In Section 4.5, we discuss various interpretations of this multiplicative factor with regard to anonymity.

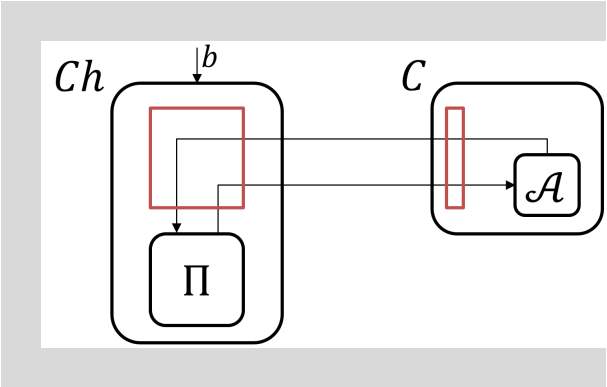

Figure 1: The structure of the AnoA framework

In this section we define and describe the separate parts of the AnoA framework, i.e., the three main components (protocol, challenger and adversary) and their interactions.

4.1 Protocol Model

Anonymous communication (AC) protocols are distributed protocols that enable multiple users to anonymously communicate with multiple recipients. Formally, an AC protocol is an interactive Turing machine.[1] We associate a protocol with a user space $\mathcal{U}$ and a recipient space $\mathcal{R}$. Users' actions are modeled as an input to the protocol and represented in the form of an ordered *input action*. Each input action contains a sender $S \in \mathcal{U}$ that sends a message m to a recipient $R \in \mathcal{R}$, together with an identifier for a session `session` .

A typical adversary in an AC protocol can compromise a certain number of parties. We model such an adversary capability as static corruption: before the protocol execution starts $\mathcal{A}$ may decide which parties to compromise.

Our protocol model is sufficiently generic for capturing multi-party protocols in classical simulation-based composability frameworks, such as UC (Canetti, 2013), GNUC (Hofheinz and Shoup, 2013), IITM (Küsters and Tuengerthal, 2013), RSIM (Backes et al., 2007), or even time-sensitive variants, such as TUC (Backes et al., 2014b) (e.g., for capturing timing-related attacks). In particular, our protocol model comprises ideal functionalities, trusted machines that are used in simulation-based composability frameworks to define security. It is straightforward to construct a wrapper for such an ideal functionality of an AC protocol that translates input tables to the expected input

[1] We stress that using standard methods, a distributed protocol with several parties can be represented by one interactive Turing machine.

of the functionality. We present such a wrapper for Tor in Section 7.

We assume that the protocol uses session IDs `session` to coordinate and relate sessions of users with each other, i.e., that it tells the challenger for each message (or each meta-data), from which session this was derived.

Moreover, we assume that the protocol provides two functions validateSender() for validating that a given term S describes a valid sender and validateRecipient() for validating that a given tuple (R, m) describes a valid recipient and a valid message for this recipient.

4.2 Challenger

In AnoA, we model anonymity as a challenge-response game between a challenger that simulates the protocol and an adversary that tries to de-anonymize users. The challenger $\texttt{Ch}(\Pi, \alpha, n, b)$ allows the adversary to adaptively control user communication in the network, up to an uncertainty of one bit for challenges, and is parametric in the following parts: (i) the AC protocol Π to be analyzed, (ii) the so called *anonymity function* α, that describes the specific variant of anonymity we are interested in (sender anonymity, recipient anonymity, relationship anonymity), the number of challenges n the adversary is allowed to issue, and the challenge bit b which determines the decision the challenger takes in challenge inputs from the adversary. The implementation of `Ch` is illustrated in Figure 2. In the following we describe `Ch` in detail.

Input Messages. On regular inputs of the form $(\texttt{Input}, r = (S, R, m, \texttt{session}), \texttt{ID})$ from the adversary, `Ch` runs the protocol Π with input $(r = (S, R, m, \texttt{session}), \texttt{ID})$ without altering it.

Challenge Messages. On challenges of the form $(\texttt{Challenge}, r_0, r_1, (\texttt{Ch}, \Psi))$, `Ch` first checks the validity of the challenge before choosing the input r_b based on its challenge bit b. This validity check on the side concerns the form of r_i: it has to contain a valid sender (using validateSender()) and a valid message-recipient pair (using validateRecipient()).

On the other side, the validity of the challenge tag Ψ has to be checked: First, it is checked whether $\Psi \in [1 \dots n]$ to make sure that the adversary does not issue too many challenges. After that the state of the challenge with tag Ψ has to be determined: let T be the set of all active challenges. If the adversary has not yet issued a challenge with tag Ψ, the state is set to `FRESH CHALLENGE` and the challenge Ψ is added to T. If Ψ is already active, i.e. $\Psi \in T$, the challenger checks whether it is already finished (indicated by the `CHALLENGE OVER` state): if this is the case, the challenger returns $\perp$ to the adversary to indicate an invalid input.

If the challenge is valid, `Ch` runs α on the challenge inputs r_0 and r_1 together with `Ch`'s challenge bit b and the current state of the challenge Ψ. The output of α is then used as the input in which Π is run.

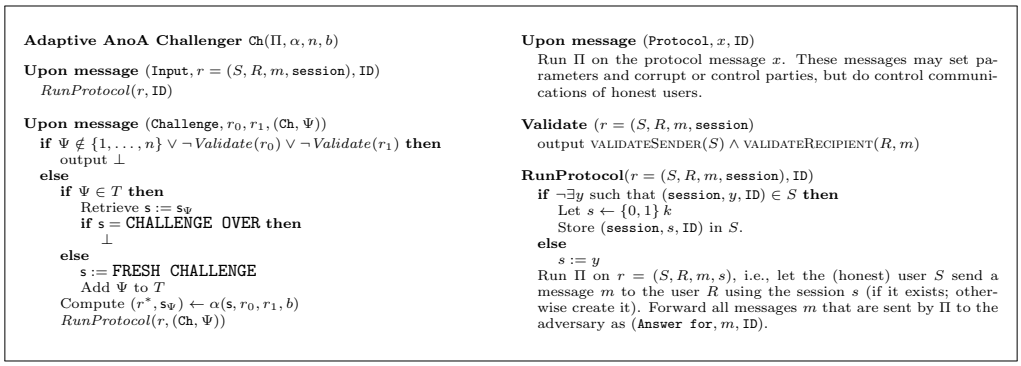

Figure 2: Adaptive AnoA Challenger

We use challenge tags to allow the adversary to interleave several challenges that are within different stages at the same time: e.g., the adversary might first start an unlinkability challenge (with tag Ψ_1). Before this challenge is `CHALLENGE OVER`, the adversary might start another challenge (with another tag Ψ_2) or continue the first one (with Ψ_1).

The idea behind distinguishing challenge sessions and other sessions is to avoid that the adversary exploits the session mechanics for getting an unfair advantage. Consider the following example: in a sender anonymity game, the adversary sends a challenge that lets either Alice or Bob start a session. Then, the adversary instructs Alice to send another message with the same session ID. For many protocols, it is trivially distinguishable, whether two messages are sent within one session or in separate sessions.

Protocol Messages. In addition to input and challenge messages, the adversary can also send protocol messages of the form $(\mathtt{Protocol}, x, \mathtt{ID})$. The protocol messages contain meta messages sent to the protocol that allow the adversary, e.g., to compromise network parties or network links and set other parameters of the protocol Π.

4.3 Adversary Model

In AnoA, the adversary $\mathcal{A}$ is modeled as a `ppt` TM that interacts with the challenger $\mathtt{Ch}(\Pi, \alpha, n, b)$ described in Section 4.2. Together with regular input messages, the adversary is allowed to issue up to n challenges that conform with the requirements set by the anonymity function α. The goal of $\mathcal{A}$ is to break the anonymity provided by Π by correctly guessing the challenge bit b used by `Ch` in computing α.

The adversary receives protocol output from the challenger, which includes all messages it is able to intercept or see throughout the protocol execution. Depending on the definition of Π, $\mathcal{A}$ can additionally send protocol messages to Π that allow $\mathcal{A}$

to compromise or control parties in the network, which models that the adversary can (partially) compromise the network. To model adaptive scenarios, e.g., for protocols in which users answer to messages, the adversary can adaptively construct its inputs to the challenger based on the history of outputs observed.

Adversary Restrictions for Modeling Specific Scenarios. The amount of information gained by the adversary completely depends on the leakage modeled by the protocol Π simulated by $\texttt{Ch}$. However, we might want to vary the adversary's capabilities without modifying the protocol, in order to provide anonymity guarantees not only for the strongest possible, but also a possibly weaker adversary, e.g., by reducing the amount of information the adversary has access to or the amount of control it has over the challenges.

To achieve this, we introduce the notion of adversary classes C: these adversary classes are essentially wrappers around the actual adversary that allow us to filter the inputs the adversary sends to the challenger, as well as the outputs of the protocol the adversary receives. In the following we will assume that we always have an adversary class C wrapped around the actual adversary $\mathcal{A}$ and denote the resulting new adversary with $\mathrm{C}(\mathcal{A})$. The original case without adversary classes can obviously be modeled by having an adversary class that simply forwards all inputs and outputs without filtering them.

In Section 6.2, we formalize the restriction of adversaries with such adversary classes and present an important sanity condition (which we call single-challenge reducibility). Moreover, we show a generic template for adversary classes for which we prove that it always satisfies all desired properties (called Plug'n'Play adversary classes).

4.4 Anonymity

In AnoA, we quantify the anonymity provided by the protocol Π simulated by $\texttt{Ch}(\Pi, \alpha, n, b)$ in terms of the advantage the adversary $\mathrm{C}(\mathcal{A})$ has in correctly guessing $\texttt{Ch}$'s challenge bit b. We measure this advantage in terms of indistinguishability of random variables (both additively and multiplicatively), where the random variables in question represent the output of the interactions $\langle \mathrm{C}(\mathcal{A}) | \texttt{Ch}(\Pi, \alpha, n, 0) \rangle$ and $\langle \mathrm{C}(\mathcal{A}) | \texttt{Ch}(\Pi, \alpha, n, 1) \rangle$.

This approach is motivated by similar definitions presented under different contexts, e.g. differential indistinguishability (Backes et al., 2015a) for cryptography with imperfect randomness, differential privacy (Dwork, 2008) for privacy in statistical databases. Formally, we say that the protocol Π provides $(\alpha, n, \epsilon, \delta)$-$\texttt{IND-ANO}$ if the indistinguishability between the random variables defined by $\langle \mathrm{C}(\mathcal{A}) | \texttt{Ch}(\Pi, \alpha, n, 0) \rangle$ and $\langle \mathrm{C}(\mathcal{A}) | \texttt{Ch}(\Pi, \alpha, n, 1) \rangle$ can be bounded by the multiplicative factor ϵ and the additive factor δ.

Definition 1 ((n, ϵ, δ)-$\texttt{IND-ANO}$)**.** *A protocol* Π *is* $(n, \epsilon, \delta, \alpha)$-$\texttt{IND-ANO}$ *for a class of adversaries* C*, with* $\varepsilon \geq 0$ *and* $0 \leq \delta \leq 1$*, if for all* $\texttt{ppt}$ *machines* $\mathcal{A}$*,*

$$\Pr\left[0 = \langle \mathrm{C}(\mathcal{A}(n)) | \texttt{Ch}(\Pi, \alpha, n, 0) \rangle\right] \leq e^{n\varepsilon} \Pr\left[0 = \langle \mathrm{C}(\mathcal{A}(n)) | \texttt{Ch}(\Pi, \alpha, n, 1) \rangle\right] + e^{n\varepsilon} n\delta.$$

Anonymity Function. The anonymity guarantee we provide in AnoA is parametric in the anonymity function α. This function encodes the specific anonymity notion we analyze, e.g. sender anonymity, recipient anonymity, or relationship anonymity. The main purpose of the anonymity function α is to validate challenges issued by the adversary for the anonymity notion represented by α, and choose one of the challenge inputs based on the challenger $\texttt{Ch}$'s challenge bit (cf. Section 4.2). We construct the anonymity functions for different anonymity notions in Section 5.

4.5 Interpretation of Anonymity Guarantees

In Definition 1, we bound the distinguishability of $\langle \mathrm{C}(\mathcal{A}|\texttt{Ch}(\Pi, \alpha, n, 0)\rangle$ and $\langle \mathrm{C}(\mathcal{A}(n))|\texttt{Ch}(\Pi, \alpha, n, 1)\rangle$ by the additive factor δ and the multiplicative factor ε. In the following we discuss suitable interpretations of these parameters.

Additive Factor. The additive factor δ gives us the probability with which the adversary can actually distinguish both $\texttt{Ch}(\Pi, \alpha, n, 0)$ and $\texttt{Ch}(\Pi, \alpha, n, 1)$. Classic differential indistinguishability notions usually require that δ is negligible. However, these previous notions also do not allow for an adversary that actively compromises the mechanism in question, which we do require for the analysis of anonymous communication networks: the analysis of AC protocols under partial compromisation naturally leads to a non-negligible δ, which can be interpreted as the likelihood of a *distinguishing event* which allows the adversary to break anonymity and identify the hiding party. In the analysis of Tor in Section 7, our main effort will be put towards identifying such distinguishing events and determining their likelihood.

Multiplicative Factor. Following the discussion of the multiplicative factor and its impact on cryptographic guarantees presented by Backes, Kate, Meiser and Ruffing (Backes et al., 2015a), we now discuss its impact in terms of anonymity.[2] We refer the interested reader to their work for more details. The multiplicative factor ε in Definition 1 allows for qualitatively different analyses than a strictly additive advantage of the adversary. It indicates the distinguishability of both random variables, i.e. $\texttt{Ch}(\Pi, \alpha, n, 0)$ and $\texttt{Ch}(\Pi, \alpha, n, 1)$, under ideal conditions: it quantifies the leakage inherent to the analyzed protocol Π, even against a passively observing adversary that does not compromise any parts of the network but only knows of in- and out-going messages.

If the adversary has an advantage δ in distinguishing between two scenarios, e.g., two possible recipients of a message, this might stem from a chance of breaking the cryptography involved in the protocol, i.e., by guessing a decryption key, or a chance to entirely compromise relevant protocol parties. Generally, such an advantage can stem from a certain probability of finding hard evidence or proof for the scenario. In contrast, a (purely) multiplicative factor describes that the adversary has some advantage in guessing the scenario, but is provably not able to find hard evidence or even proof. Thus,

[2] Backes et al. (2015a) also show that such a multiplicative e^{ε} can always be described in terms of the additive δ. The converse, however, does not hold.

if by introducing a relatively small multiplicative e^{ε} we can reduce δ accordingly, the guarantee is more tight.

Example 1 (Multiplicative factor for recipient anonymity)**.** *Building upon AnoA, Backes et al. (2014a) have shown that a multiplicative factor indeed helps to capture non-distinguishing events for Tor (without the entry guard mechanics). They show that especially for recipient anonymity, if different recipients require different ports, even without compromised nodes Tor is slightly vulnerable to ISPs. However, this vulnerability completely falls within ε.*

The reason for this observation is as follows. An ISP can always see the entry node. If two recipients with different ports are selected, different subsets of nodes can be chosen as exit nodes. This discrepancy translates to different probabilities for the entry nodes as well (as exit nodes are chosen first and the choice of entry nodes slightly depends on the chosen exit node), resulting in a multiplicative factor.

The analysis of the Tor protocol we present in Section 7 is very basic and thus only considers a multiplicative factor of $\varepsilon = 0$. More involved analyses that take into account Tor's path selection and adversarial infrastructure, and thus produces anonymity guarantees with non-zero multiplicative factors, can be found in follow up work by Backes et al. (Backes et al., 2014a; 2015b).

This concludes the introduction of the AnoA framework. In the subsequent section we present the formalization of various anonymity notions in the AnoA framework.

5 Anonymity Notions

In this section we present a variety of anonymity notions for AnoA. These notions are represented as *anonymity functions* that describe the challenges in the game-based definition between our challenger (see Figure 2) and the adversary. First we describe rather simple anonymity functions for challenges consisting of individual messages, before describing more complex notions (e.g., for communication sessions).

5.1 Single-Message Anonymity

Many protocols consider anonymity on a per-message basis. For such protocols, we define anonymity in terms of individual messages. As we show in Section 5.3, this in not a particularly strong restriction, since anonymity for an individual challenge (i.e., for one message) implies anonymity for several messages, for a large class of adversaries.

Sender Anonymity

Sender anonymity characterizes the anonymity of users against a malicious server. In contrast to other notions from the literature, we define sender anonymity as the inability of an observer to decide which of two *self-chosen* users have been communicating.

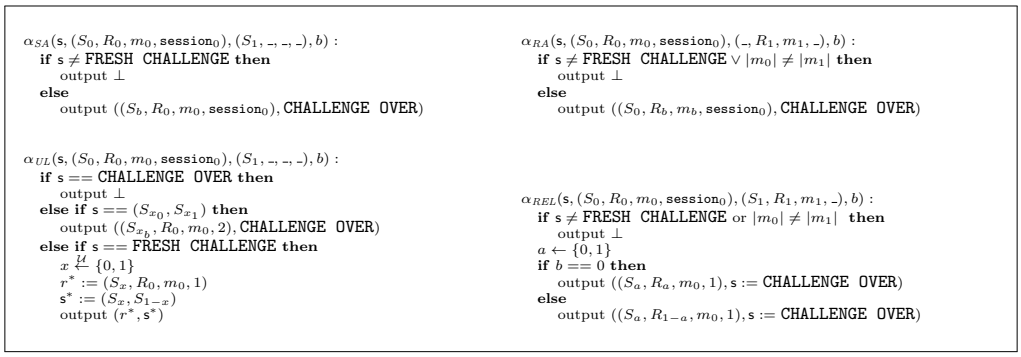

Figure 3: The (single-message) anonymity functions.

We formalize our notion of sender anonymity with the definition of an anonymity function α_{SA} as depicted in Figure 3. Basically, α_{SA} selects one of two possible challenge users and makes sure that the users cannot be distinguished by the chosen recipient(s) or message(s).

Definition 1 (Sender anonymity)**.** *A protocol* Π *provides* (ε, δ)-sender anonymity *if it is* $(1, \alpha_{SA}, \varepsilon, \delta)$-IND-ANO *for* α_{SA} *as defined in Figure 3.*

Example 2 (Sender anonymity)**.** *The adversary* $\mathcal{A}$ *decides that it wants to use users Alice and Bob in the sender anonymity game. It sends a challenge* (Challenge, (*Alice, evilserver.com, m,* session), (*Bob, evilserver.com, m,* session), 1) *for sending some message* m *of* $\mathcal{A}$*'s choice to a (probably corrupted) recipient evilserver.com. The anonymity function* α_{SA} *now chooses (depending on the challenge bit) one of the two senders and outputs the corresponding command to the protocol.*

Recipient Anonymity

Recipient anonymity characterizes that the recipient of a communication remains anonymous, even to observers that have knowledge about the sender in question. Our notion of recipient anonymity is defined analogously to sender anonymity: the anonymity function selects one of two possible recipients for a message. To account for possibly different message requirements for different recipients but to still counter attacks based on the pattern of those messages, the anonymity function checks whether the messages have the same length.

Definition 2. *A protocol* Π *provides* (ε, δ)-recipient anonymity *if it is* $(1, \alpha_{RA}, \varepsilon, \delta)$-IND-ANO *for* α_{RA} *as defined in Figure 3.*

Trivial Attacks Against Recipient Anonymity. In a website fingerprinting attack, the attacker checks traffic features (e.g., volume and direction changes) of the response, which

in many cases is unique as recent studies show (Panchenko et al., 2011; Cai et al., 2012). Such a trivial attack is possible against any low-latency anonymous communication protocol and should hence be excluded for quantifying the anonymity guarantees of a low-latency anonymous communication protocol. Thus, we suggest restricting the responses to "similar responses" (i.e., responses of the same length and pattern) that do not trivially identify the recipient of the initial message. In Section 7.2 we describe how we restrict such responses for our analysis of the Tor protocol.

Sender Unlinkability

Sender unlinkability characterizes that it is hard to distinguish two messages sent by the same sender from two messages sent by different senders. A protocol satisfies *sender unlinkability* if for any two messages, the adversary cannot determine whether these messages are sent by the same user (Pfitzmann and Hansen, 2010). We require that the adversary does not know whether two challenge messages come from the same user or from different users. We formalize this intuition by letting the adversary send two challenge messages for one challenge. The first challenge message defines the possible senders of the messages. From those, one (random) sender is chosen and stored as the active sender, while the other sender is stored as alternative sender. For the second challenge message, depending on the challenge bit, either the active sender of the challenge sends the second message, or the alternative sender. This is formalized in the anonymity function α_{UL} as defined in Figure 3.

As before, we say a protocol Π fulfills sender unlinkability, if no adversary $\mathcal{A}$ can sufficiently distinguish $\texttt{Ch}(\Pi, \alpha_{UL}, 0)$ and $\texttt{Ch}(\Pi, \alpha_{UL}, 1)$. This leads to the following concise definition.

Definition 3 (Sender unlinkability)**.** *A protocol* Π *provides* (ε, δ)-sender unlinkability *if it is* $(1, \alpha_{UL}, \varepsilon, \delta)$-`IND-ANO` *for* α_{UL} *as defined in Figure 3.*

Example 3 (Sender unlinkability)**.** *The adversary* $\mathcal{A}$ *decides that it wants to use users Alice and Bob in the unlinkability game. It sends a challenge message* $(\texttt{Challenge}, (\mathit{Alice}, m, R, \texttt{session}), (\mathit{Bob}, m, R, \texttt{session}), \Psi)$*. The anonymity function randomly selects one of the senders, say, Bob, and instructs him to send message* m *to the recipient* R*. So far, the adversary cannot break unlinkability, as only one random sender was used.*

Later, $\mathcal{A}$ *sends another challenge message* $(\texttt{Challenge}, (\_, m, R, \texttt{session}), (\_, \_, \_), \Psi)$ *for the same challenge* Ψ*. Now the behavior is different for different challenge bits: If the challenge bit* b *is zero, the anonymity function will reuse the same sender as for the first message (Bob). Otherwise, it will instruct Alice to send the second message.*

Relationship Anonymity

A protocol satisfies *relationship anonymity*, if for any communication, the adversary cannot determine sender and recipient of this communication at the same time (Pfitzmann and Hansen, 2010). We model this property by letting the anonymity function α_{REL} choose one of two possible senders and one of two possible recipients for a given challenge.

We then group the four possible combinations in two sets (one defined by the adversary and one modified by the anonymity function) and let the adversary guess from which set the communication came.

More formally, the anonymity function α_{REL}, as defined in Figure 3, chooses either one of the given challenge actions of the adversary (for $b = 0$) or creates a modified version of one of the challenge actions (for $b = 1$): it selects the sender of one action, but message and recipient from another action. Moreover, the anonymity function makes sure that the possible messages are of the same length (to exclude obvious distinctions).

We say that Π fulfills relationship anonymity, if no adversary can sufficiently distinguish $\texttt{Ch}(\Pi, \alpha_{REL}, 1, 0)$ and $\texttt{Ch}(\Pi, \alpha_{REL}, 1, 1)$.

Definition 4 (Relationship anonymity)**.** *A protocol* Π *provides* (ε, δ)-relationship anonymity *if it is*
$(1, \alpha_{REL}, \varepsilon, \delta)$-`IND-ANO` *for* α_{REL} *as defined in Figure 3.*

Example 4 (Relationship anonymity)**.** *The adversary* $\mathcal{A}$ *decides that it wants to use users Alice and Bob and the recipients Charly and Eve in the relationship anonymity game. It wins the game if it can distinguish between the scenario "0" where Alice sends* m_1 *to Charly or Bob sends* m_2 *to Eve and the scenario "1" where Alice sends* m_2 *to Eve or Bob sends* m_1 *to Charly. Only one of those four possible instructions will be fed to the protocol.*

$\mathcal{A}$ *sends a challenge message* $(\texttt{Challenge}, (Alice, Charly, m, \texttt{session}), (Bob, Eve, m, \texttt{session}), \Psi)$ *to* `Ch`*. The anonymity function* α_{REL} *then randomly selects a sender* $a \in \{Alice, Bob\}$*, say Alice. Then, depending on the challenge bit b, the function outputs either* $(Alice, Charly, m_1, \texttt{session})$ *as specified by* $\mathcal{A}$ *(for* $b = 0$*), or* $(Alice, Eve, m_2, \texttt{session})$ *as a modified instruction (for* $b = 1$*).*

5.2 Anonymity Function Soundness

In this section, we relate the (single-message) anonymity notions defined above to classic definitions directly derived the seminal work by Pfitzmann and Hansen (2010). We examplarily show that for sender anonymity and sender unlinkability, we can translate our definitions to the classic definitions (and vice versa), whereas for relationship anonymity we discuss crucial differences and their implications for the respective anonymity notions.

Sender Anonymity

The notion of sender anonymity is introduced in Pfitzmann and Hansen (2010) as follows:

> Anonymity of a subject from an adversary's perspective means that the adversary cannot sufficiently identify the subject within a set of subjects, the anonymity set.

From this description, we formalize their notion of sender anonymity in the anonymity function ${\alpha_{SA}}^c$ shown in Figure 4. This anonymity function then allows us to define the

notion of δ-sender anonymity, which states that, given a message m sent to a recipient R, the advantage of the adversary to guess the right sender from the user space $\mathcal{U}$ can be bound by δ.

Definition 5 (δ-sender anonymity)**.** *A protocol* Π *with user space* $\mathcal{U}$ *of size* N *has* δ-sender anonymity *if for all* ppt*-adversaries* $\mathcal{A}$

$$\Pr\left[u^* = u \mid u^* = \langle \mathcal{A} | \mathtt{Ch}(\Pi, {\alpha_{SA}}^c, 1, u)\rangle, u \xleftarrow{\mathcal{U}} \mathcal{U}\right] \leq \frac{1}{N} + \delta,$$

where the anonymity function ${\alpha_{SA}}^c$ *as defined as in Figure 4.*

This definition is quite different from our own definition with the anonymity function α_{SA}. While α_{SA} requires $\mathcal{A}$ to simply distinguish between two possible outcomes, Definition 5 requires $\mathcal{A}$ to correctly guess the right user from the user space $\mathcal{U}$. Naturally, α_{SA} is stronger than the definition above. Indeed, we can quantify the gap between the definitions: Lemma 1 states that an AC protocol satisfying $(0, \delta)$-sender anonymity implies that this AC also provides δ-sender anonymity.

Lemma 1 (Sender anonymity)**.** *For all protocols* Π *over a (finite) user space* $\mathcal{U}$ *of size* N *it holds that if* Π *provides* $(0, \delta)$*-sender anonymity, then* Π *also provides* δ*-sender anonymity as in Definition 5.*

Proof Outline. We show the contraposition of the lemma: an adversary $\mathcal{A}$ that breaks sender anonymity, can be used to break IND-ANO for α_{SA}. We construct an adversary B against IND-ANO for α_{SA} by choosing the senders of the challenge rows at random, running $\mathcal{A}$ on the resulting game, and outputting the same as $\mathcal{A}$. For $\mathcal{A}$ the resulting view is the same as in the sender anonymity game; hence, B has the same success probability in the IND-ANO game as $\mathcal{A}$ in the sender anonymity game. □

In the converse direction, we lose a factor of $\frac{1}{N}$ in the reduction, where N is the size of the user space. If an AC protocol Π provides δ-sender anonymity, we only get $(0, \delta \cdot N)$-sender anonymity for Π.

Lemma 2. *For all protocols* Π *over a (finite) user space* $\mathcal{U}$ *of size* N *it holds that if* Π *provides* δ*-sender anonymity, then* Π *also provides* $(0, \delta \cdot N)$*-sender anonymity.*

Proof Outline. We show the contraposition of the lemma: an adversary $\mathcal{A}$ that breaks IND-ANO for α_{SA}, can be used to break δ-sender anonymity. We construct an adversary B against sender anonymity by running $\mathcal{A}$ on the sender anonymity game and outputting the same as $\mathcal{A}$. If the wishes of $\mathcal{A}$ for the challenge senders coincide with the sender that the challenger chose at random, the resulting view is the same as in the IND-ANO game for α_{SA}; hence, B has a success probability of δ/N in the sender anonymity game if $\mathcal{A}$ has a success probability of δ in the IND-ANO game for α_{SA}. □

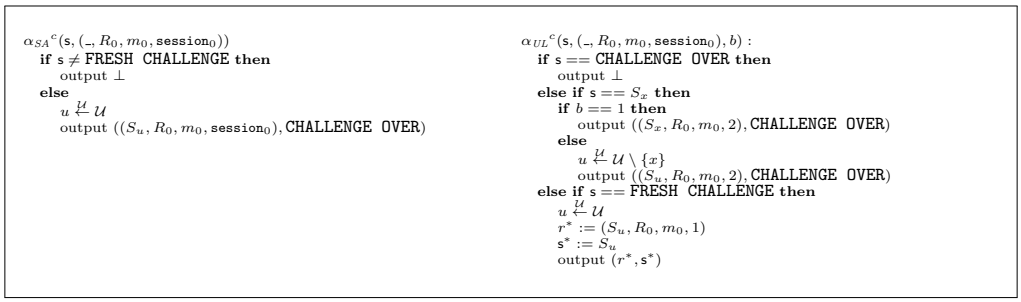

Figure 4: The anonymity functions for the classic anonymity definitions.

Unlinkability

The notion of unlinkability is defined in Pfitzmann and Hansen (2010) as follows:

> Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an adversary's perspective means that within the system (comprising these and possibly other items), the adversary cannot sufficiently distinguish whether these IOIs are related or not.

Again, we formalize this notion in our model. We leave the choice of potential other items in the system completely under adversary control. Also, the adversary controls the "items of interest" (IOI) by choosing when and for which recipient/messages he wants to try to link the IOIs. Formally, we define a game between a challenger $\mathtt{Ch}(\Pi, \alpha_{UL}{}^c, 1, b)$, using the anonymity function $\alpha_{UL}{}^c$ depicted in Figure 4, and an adversary $\mathcal{A}$ as follows: among regular message inputs, $\mathcal{A}$ can send a first challenge that contains the first message of the unlinkability challenge to be sent through the network. The anonymity function $\alpha_{UL}{}^c$ then chooses one (random) user u as the sender and sends the message through the network, while remembering this sender in its state.

At a later point, the $\mathcal{A}$ then sends the second challenge, which contains the second message that either is or is not linked to the first challenge message. The anonymity function, based on the challenge bit b, uses the same user u as the sender of the second message or chooses a new user $u' \neq u$ as the second sender. The adversary wins the game if he is able to distinguish whether the same user was chosen for both challenges (i.e. the IOIs are linked).

Definition 6 (δ-sender unlinkability)**.** *A protocol* Π *with user space* $\mathcal{U}$ *has* δ-sender unlinkability *if for all* ppt*-adversaries* $\mathcal{A}$

$$\left|\Pr\left[b = 0 : b \leftarrow \langle \mathcal{A} | \mathtt{Ch}(\Pi, \alpha_{UL}{}^c, 1, 0)\rangle\right] - \Pr\left[b = 0 : b \leftarrow \langle \mathcal{A} | \mathtt{Ch}(\Pi, \alpha_{UL}{}^c, 1, 1)\rangle\right]\right| \leq \delta$$

where the anonymity function $\alpha_{UL}{}^c$ *is as defined in Figure 4.*

We show that our notion of sender unlinkability using the adjacency function α_{UL} is much stronger than the δ-sender unlinkability definition above: $(0, \delta)$-sender unlinkability for an AC protocol directly implies δ-sender unlinkability; we do not lose any anonymity.

Lemma 3 (sender unlinkability)**.** *For all protocols* Π *over a user space* $\mathcal{U}$ *it holds that if* Π *has* $(0, \delta)$*-sender unlinkability,* Π *also has* δ*-sender unlinkability as in Definition 6.*

Proof Outline. We show the contraposition of the lemma: an adversary $\mathcal{A}$ that breaks sender unlinkability can be used to break `IND-ANO` for α_{UL}. We construct an adversary B against `IND-ANO` for α_{UL} by choosing the senders of the challenge rows at random, running $\mathcal{A}$ on the resulting game, and outputting the same as $\mathcal{A}$. For $\mathcal{A}$ the resulting view is the same as in the sender unlinkability game; hence, B has the same success probability in the `IND-ANO` game for α_{UL} as $\mathcal{A}$ in the sender unlinkability game. □

For the converse direction, however, we lose a factor of roughly N^2 for our δ. Similar to the sender anonymity case, showing that a protocol provides δ-sender unlinkability only implies that the protocol provides $(0, \delta \cdot N(N-1))$-sender unlinkability.

Lemma 4 (sender unlinkability)**.** *For all protocols* Π *over a user space* $\mathcal{U}$ *of size* N *it holds that if* Π *has* δ*-sender unlinkability as in Definition 6,* Π *also has* $(0, \delta \cdot N(N-1))$*-sender unlinkability.*

Proof Outline. We show the contraposition of the lemma: an adversary $\mathcal{A}$ that breaks `IND-ANO` for α_{UL}, can be used to break sender unlinkability. We construct an adversary B against sender unlinkability by running $\mathcal{A}$ on the sender unlinkability game and outputting the same as $\mathcal{A}$. If the senders from the challenge from of $\mathcal{A}$ coincide with the senders that the challenger chose at random, the resulting view is the same as in the `IND-ANO` game for α_{UL}; hence, B has a success probability of $\delta / N(N-1)$ in the sender unlinkability game if $\mathcal{A}$ has a success probability of δ in the `IND-ANO` game for α_{UL}. □

Relationship Anonymity

While for sender anonymity and sender unlinkability our notions coincide with the definitions used in the literature, we find that for relationship anonymity, many of the interpretations from the literature are not accurate. In their Mixnet analysis, Shmatikov and Wang (2006) define relationship anonymity as 'hiding the fact that party A is communicating with party B'. Feigenbaum et al. (2007b) also take the same position in their analysis of the Tor network. However, in the presence of such a powerful adversary, as considered in this work, these previous notions collapse to recipient anonymity since they assume knowledge of the potential senders of some message.

We consider the notion of relationship anonymity as defined in Pfitzmann and Hansen (2010): the anonymity set for a message m comprises the tuples of possible senders and recipients; the adversary wins by determining which tuple belongs to m. However, adopting this notion directly is not possible: an adversary that gains partial information

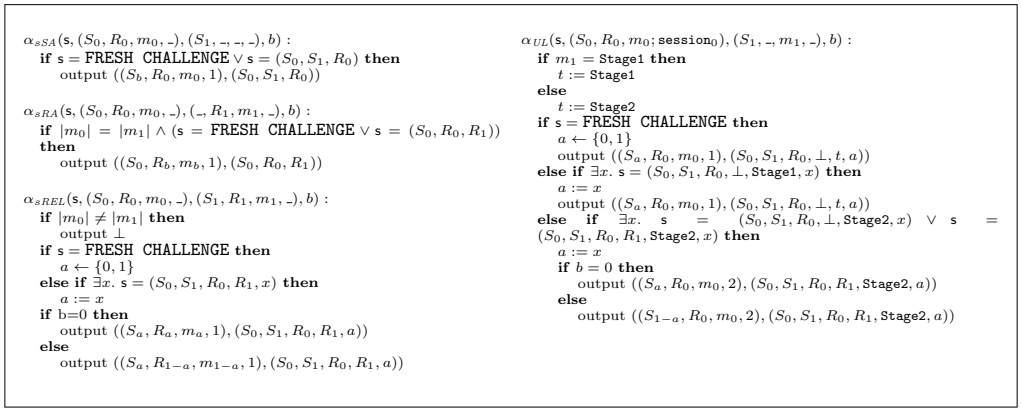

Figure 5: The session anonymity functions.

(e.g. if he breaks sender anonymity), also breaks the relationship anonymity game, all sender-recipient pairs are no longer equally likely. Therefore we think that approach via the adjacency function gives a better definition of relationship anonymity because the adversary needs to uncover both sender and recipient in order to break anonymity.

5.3 Session Anonymity Notions

Some anonymous communication protocols handle messages from users not individually, but in sessions consisting of a multitude of messages, without significantly harming the anonymity. For such protocols (such as Tor), we define variants of the anonymity notions from above for sessions. A pseudocode description of the anonymity functions can be found in Figure 5.

Note that these notions allow for optimized guarantees, i.e., guarantees that are stronger than the ones derived by transforming single-message games guarantees into multiple-message games in a generic manner (see Theorem 1 in Section 6.2 for this generic result).

Session Sender Anonymity. For session sender anonymity, we make sure that all messages that belong to the same challenge have the same sender. The challenger makes sure that they additionally have the same session ID. We do not restrict the number of messages per challenge (but the protocol might restrict the number of messages per session).

Session Recipient Anonymity. For session recipient anonymity, we make sure that all messages that belong to the same challenge have the same sender, and for single-message recipient anonymity we check that both messages have the same length.

Session Relationship Anonymity. Relationship anonymity describes the anonymity of a relation between a user and the recipients, i.e., an adversary cannot find out both the sender of a message and the recipient.

Session Sender Unlinkability. Our notion of session sender unlinkability allows the adversary (for every challenge) to first run arbitrarily many messages in one session (`Stage1`) and then arbitrarily many messages in another session (`Stage2`). In each session there is only one sender used for the whole session (that is chosen at random when the first challenge message for this challenge is sent), but depending on b, either the same or the other user is used for the second session.

6 Adversary Classes

AnoA assumes a strong attacker that can choose the user's inputs and—depending on the protocol—receives the responses from the servers. Moreover, the adversary can freely compromise protocol parties and networks links. Many AC protocols, such as Tor or I2P do not offer perfect anonymity and even offer only little anonymity against such powerful adversaries. However, by restricting the adversary's capabilities, we can analyze a multitude of realistic scenarios and give meaningful guarantees.

Technically, the adversary's capabilities can be restricted by constructing a wrapper around the actual adversary. Since such a wrapper that internally runs an attacker models an entire class of adversaries, we call such a wrapper an *adversary class*. In this section, we give examples for adversary classes, construct a general template for adversary classes (*Plug'n'Play* adversary classes), and identify sanity conditions for adversary classes, called single-challenge reducibility. For a single-challenge reducible adversary class it suffices to show anonymity bounds for a single challenge to immediately derive bounds for more challenges.

6.1 Defining Adversary Classes

An *adversary class* is a `ppt` wrapper that restricts the adversary $\mathcal{A}$ in two dimensions: (a) in its possible output behavior, and thus, in its knowledge about the world and (b) in its possible observations, and thus in its ability to gain insights about leakage. The following example shows a scenario in which an (unrestricted) adaptive adversary might be too strong.

Example 5 (Tor with entry guards)**.** *Consider the AC protocol Tor with so-called* entry guards*. Every user selects only one entry node (his guard) that serves as the entry node of every circuit. A new guard is chosen only after several months. As a compromised entry node is fatal for the security guarantees that Tor can provide, the concept of entry guards helps in reducing the risk of choosing a malicious node. However, if such an entry guard is compromised the impact is more severe since an entry guard is used for a large number of circuits.*

An adaptive adversary can choose its targets adaptively and thus perform the following attack. It (statically) corrupts some nodes and then sends (polynomially) many messages $(\texttt{Input}, r = (S, \_, \_, \_))$ *for different users* S*, until one of them, say Alice, chooses a compromised node as its entry guard. Then* $\mathcal{A}$ *proceeds by using Alice and some other user in a challenge. As Alice will use the compromised entry guard again, the adversary trivially wins the game.*

Although this extremely successful attack is not unrealistic (it models the fact that if there are compromised entry guards, then some users will likely use them and consequently they *will* be de-anonymized), it might not depict the attack scenario that we are interested in. Thus, we define an adversary class that makes sure that the adversary cannot choose its targets. Whenever the adversary starts a new challenge for (session) sender anonymity, the adversary class draws two users at random and places them into the challenge messages of this challenge.

Definition 7. *An* adversary class *is defined by a* `ppt` *machine* $\mathrm{C}(\cdot)$ *that encapsulates the adversary* $\mathcal{A}$*. It may filter and modify all outputs of* $\mathcal{A}$ *to the challenger in an arbitrary way, withhold them or generate its own messages for the challenger. Moreover it may communicate with the adversary.*

We say that a protocol is secure against an adversary class $\mathrm{C}(\cdot)$*, if it is secure against all machines* $\mathrm{C}(\mathcal{A})$*, where* $\mathcal{A}$ *is an arbitrary* `ppt` *machine.*

6.2 Single-Challenge Reducible Adversary Classes

In this section we show that adaptive `IND-ANO` against adversaries that only use one challenge immediately implies adaptive `IND-ANO` against adversaries that use more than one challenge. The quantitative guarantees naturally depend on the number of challenges.

For adversaries that are not restricted by an adversary class the multi-challenge generalization holds and yields similar guarantees as for differential privacy: both ε and δ increase linearly in the number of challenges, just as for differential privacy a larger distance between databases linearly increases them. The intuition behind this generalization is similar to comparable guarantees for differential privacy, as long as the protocol is oblivious to whether it was instructed by a challenge or an input message. Any additional challenge can increase the advantage of an adversary (linearly), but cannot allow the adversary to perform attacks that were impossible without this additional challenge. Intuitively, if there was a strong attack against the protocol that required $k \geq 2$ challenges, then the adversary could simulate this attack although it only sends one challenge message: The adversary randomly samples a challenge bit $b_{\mathcal{A}}$ and then simulates the first $k-1$ challenge messages by computing the anonymity function on them, using $b_{\mathcal{A}}$ as the challenge bit, and sending the resulting messages as an input messages. Finally, it sends the last challenge as in the original attack. If the guess of the challenge bit was correct, the protocol receives exactly the same messages in this 1-challenge-attack as it receives in the original k-challenge-attack. Consequently, the attack can be performed. Adversary classes, however, can treat challenge messages in a special way such that anonymity guarantees established for a single challenge tag

cannot be generalized to several challenge tags. Hence, by restricting the adversary the number of challenges (sent by the adversary) can allow for new attacks instead of linearly increasing the advantage. The next example illustrates this problem.

Example 6 (Non-single-challenge reducible adversary class)**.** *Consider the following adversary class* $\mathrm{C}_{\mathtt{nc}}$*. The adversary class relays all input messages to the challenger and replaces the messages of the first challenge by (say) random strings or error symbols. The adversary class simply relays all messages to the challenger except for the ones belonging to the first challenge.*

Every protocol, even if completely insecure, will be IND-ANO *secure for one challenge for the class* $\mathrm{C}_{\mathtt{nc}}$ *(as the adversary cannot gain any information about the bit* b *of the challenger), but it might not necessarily be secure for more than one challenge.*

Requirements for Single-Challenge Reducibility

We show that single-challenge reducibility can be achieved if the adversary can—despite potential modifications of the adversary class—cause the same protocol behavior with inputs messages as with challenge messages. More precisely, we require that the entire behavior of the adversary class, the challenger and the protocol on challenges can be simulated by an adversary that guesses the challenge bit correctly and sends inputs instead. To this end, we introduce a machine $\mathtt{S}$, called the challenge-simulator that sits between the adversary and the adversary class and that simulates challenge messages (sent by the adversary) with input messages.

Technically, we compare the normal scenario $Real^{(\mathrm{C}\leftrightarrow\mathcal{A})}$ with the challenger, the adversary class, and the adversary to the scenario $Sim^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}$ with the challenger, the adversary class, the challenge simulator, and the adversary. We consider not only the case where all challenge messages are simulated with input messages, but also all cases in which only parts of the challenge messages are simulated. Hence, we require a family of challenge simulators, indexed by a sequence z of n pairs (z_i, b_i). If $z_i = dontsimulate$, the challenge simulator $\mathtt{S}$ does not simulate the challenge messages of the ith challenge tag. If $z_i = simulate$, the challenge simulator simulates the challenge messages of the ith challenge tag as if the challenge recommendation bit b_i was the challenge bit. Construction 1 describes both scenarios in detail.

Construction 1. *Consider the following two scenarios:*

- $Real^{(\mathrm{C}\leftrightarrow\mathcal{A})}(b,n)$*:* $\mathcal{A}$ *communicates with* C *and* $\mathcal{A}$ *communicates with* $\mathtt{Ch}(b,n)$ *(as protocol). The bit of the challenger is* b *and the adversary may send challenge tags in* $\{1,\ldots,n\}$*.*
- $Sim_{\mathtt{S}_z}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(b,n)$*:* $\mathcal{A}$ *communicates with* $\mathtt{S}_z$ *that in turn communicates with* C *and* $\mathcal{A}$ *communicates with* $\mathtt{Ch}(b,n)$ *(as protocol).The bit of the challenger is* b *and the adversary may send challenge tags in* $\{1,\ldots,n\}$*.*

In the proof, we use the simulator in an intricate hybrid argument. For that argument, we additionally need that a simulation using the wrong bit for some challenges can still

simulate other challenges for the correct bit in an indistinguishable manner. For a given challenge bit b and for any pair of simulator indices z, z' the following holds: whenever z differs from z' in the simulation bit ($z_i \neq z'_i$) for some position i and $z_i = \mathit{simulate}$, the corresponding challenge recommendation bit $b_i = b$ is consistent with the challenge bit.

Definition 8 (Consistent simulator index)**.** *A* simulator index *(for n challenges) is a bit-string $z = [(z_1, b_1), \ldots, (z_n, b_n)] \in \{0,1\}^{2n}$. A pair of simulator indices $z, z' \in \{0,1\}^{2n}$ (for n challenges) is consistent w.r.t. b if*

$$\forall i \in \{1, \ldots, n\} \;\; s.t.\; z_i \neq z'_i . (z_i = \mathit{simulate} \Rightarrow b_i = b) \wedge (z'_i = \mathit{simulate} \Rightarrow b'_i = b).$$

Additionally, we require that the adversary class must not initiate challenges on its own, though it can drop challenges (*reliability*). Moreover, the adversary class is independent from the actual challenge tags per se, i.e., it does not interpret the challenge tags in a semantic way (*alpha-renaming*).

Definition 9 (Condition for adversary class)**.** *An adversary class* C *is* single-challenge reducible *for an anonymity function α, if the following conditions hold:*

1. ***Reliability:*** *$\mathrm{C}(\mathcal{A})$ never sends a message $(\texttt{Challenge}, \_, \_, \Psi)$ to the challenger before receiving a message $(\texttt{Challenge}, \_, \_, \Psi)$ from $\mathcal{A}$ with the same challenge tag Ψ.*

2. ***Alpha-renaming:*** *C does not behave differently depending on the challenge tags Ψ that are sent by $\mathcal{A}$ except for using it in its own messages $(\texttt{Challenge}, \_, \_, \Psi)$ to the challenger and in the (otherwise empty) message $(\texttt{Answer for}, \_, \Psi)$ to $\mathcal{A}$. More formally, for every permutation Π on $\mathbf{N}$, we define C^{Π} as the machine that replaces all challenge tags Ψ within messages reaching C by $\Pi(\Psi)$ and all messages sent from C containing a challenge tag Ψ by $\Pi^{-1}(\Psi)$ and that otherwise simulates C. For all such machines, C^{Π} behaves exactly as C, i.e., for each input, its outputs have the same distribution.*

3. ***Simulatability:*** *For every $n \in \mathbf{N}$ and every list $z = [(z_1, b_1), \ldots, (z_n, b_n)] \in \{0,1\}^{2n}$ there exists a machine $\mathtt{S}_z$ such that:*

   (a) *For every $i \in \{1, \ldots, n\}$, the following holds: if $z_i = \mathit{simulate}$, then $\mathtt{S}_z$ never sends a message $(\texttt{Challenge}, \_, \_, i)$ to C.*

   (b) *The games $\mathit{Real}^{(\mathrm{C} \leftrightarrow \mathcal{A})}(b, n)$ and $\mathit{Sim}_{\mathtt{S}_{\mathtt{zreal}}}^{(\mathrm{C} \leftrightarrow \mathtt{S} \leftrightarrow \mathcal{A})}(b, n)$, as defined below are computationally indistinguishable, where $\mathtt{zreal} = [(\mathit{dontsimulate}, \_), \ldots, (\mathit{dontsimulate}, \_)] \in \{0,1\}\, 2n$ for $\mathtt{S}_z$ and C.*

   (c) *for all consistent simulator indices $z, z' \in \{0,1\}^{2n}$ (see Definition 8) the scenarios $\mathit{Sim}_{\mathtt{S}_z}^{(\mathrm{C} \leftrightarrow \mathtt{S} \leftrightarrow \mathcal{A})}(b, n)$ and $\mathit{Sim}_{\mathtt{S}_{z'}}^{(\mathrm{C} \leftrightarrow \mathtt{S} \leftrightarrow \mathcal{A})}(b, n)$ (see Construction 1) are indistinguishable.*

Single-Challenge Reducibility Enables Multi-Challenge Generalization

Theorem 1. *For every protocol* Π*, every anonymity function* α*, every* $n \in \mathbf{N}$ *and every adversary class* $\mathrm{C}(\mathcal{A})$ *that is single-challenge reducible for* α*. Whenever* Π *is* $(1, \varepsilon, \delta)$-`IND-ANO` *for* $\mathrm{C}(\mathcal{A})$*, with* $\varepsilon \geq 0$ *and* $0 \leq \delta \leq 1$*, then* Π *is* $(n, n \cdot \varepsilon, n \cdot e^{n\varepsilon} \cdot \delta)$-`IND-ANO` *for* $\mathrm{C}(\mathcal{A})$*.*

Proof outline. We show the composability theorem by induction over n. We assume the theorem holds for n and compute the anonymity loss between the games $Real^{(\mathrm{C} \leftrightarrow \mathcal{A})}(0, n+1)$, where we have $b = 0$ and $Real^{(\mathrm{C} \leftrightarrow \mathcal{A})}(1, n+1)$, where we have $b = 1$ via a transition of indistinguishable, or differentially private games.

We start with $Real^{(\mathrm{C} \leftrightarrow \mathcal{A})}(0, n+1)$ and introduce a simulator that simulates one of the challenges for the correct bit $b = 0$. We apply the induction hypothesis for the remaining n challenges (this introduces an anonymity loss of $(n \cdot \varepsilon, n \cdot e^{n\varepsilon} \cdot \delta)$). The simulator still simulates one challenge for $b = 0$, but the bit of the challenger is now $b = 1$. We then simulate all remaining n challenges for $b = 1$ and thus introduce a game in which all challenges are simulated. As the bit of the challenger is never used in the game, we can switch it back to $b = 0$ again and remove the simulation of the first challenge. We can apply the induction hypothesis again (we lose (ε, δ)) and switch the bit of the challenger to $b = 1$ again. In this game, we have one real challenge (for $b = 1$) and n simulated challenges (also for $b = 1$). Finally, we remove the simulator again and yield $Real^{(\mathrm{C} \leftrightarrow \mathcal{A})}(1, n+1)$.

We refer to Section 8 for a formal proof. □

6.3 Plug'n'Play Adversary Classes

Proving single-challenge reducibility can be tedious and lengthy. Therefore, we develop a template for adversary classes (the *Plug'n'Play* template) and show that every adversary class that can be expressed in this template is single-challenge reducible.

The Plug'n'Play (PnP) template forwards all input messages from the adversary and starts for every challenge tag a new submachine, called a *challenge machine*, with a fresh memory. In particular, none of the challenge machines shares memory with any other part of the adversary class. The template only allows these challenge machines to send messages to the challenger. Additionally, the template includes a provision for blocking messages in both directions via a so-called *filter* machine that generalizes limitations on compromisation such as a budget adversary (Backes et al., 2015b). In the direction from the adversary to the challenger, the filter machine only is applied to `Protocol` messages, not to input messages or challenge messages. Given such a challenge machine M and a filter F, we construct a PnP adversary class $\mathrm{PAC}_{M,F}$ as follows.

Definition 10 (Plug'n'Play Adversary Class)**.** *For two* `ppt` *machines* M*, the* challenge machine*, and* F*, the* filter*, we define the* Plug'n'Play adversary class $\mathrm{PAC}_{M,F}$ *as follows.*

1. *Initialize* F*.*

2. *Whenever $\mathcal{A}$ sends a message $(\texttt{Protocol}, x)$, simulate F on x until it halts and outputs a message m' or an error symbol $\perp$. If it outputs a message m', send $(\texttt{Protocol}, m')$ to the challenger.*

3. *Whenever $\mathcal{A}$ sends a message $(\texttt{Input}, r, (\mathcal{A}, x))$ for any values r and x, relay this message to the challenger.*

4. *Whenever $\mathcal{A}$ sends a message $m = (\texttt{Input}, r, (\texttt{Ch}, \Psi))$ or $m = (\texttt{Challenge}, r_0, r_1, (\texttt{Ch}, \Psi))$, run M_Ψ on m, as below.*

5. *Whenever $\texttt{Ch}$ sends a message $m = (\texttt{Answer for}, x, id)$, first simulate F on $(\texttt{Answer for}, x, \perp)$ until it outputs a message $m' = (\texttt{Answer for}, x', \_)$, or an error symbol $\perp$. Then proceed as follows:*

   - *If $id == (\texttt{Ch}, \Psi)$ for some (known) value Ψ, run M_Ψ on $(\texttt{Answer for}, x', \perp)$, as described below.*
   - *Otherwise $id == (\mathcal{A}, N)$; forward $(\texttt{Answer for}, x', (\mathcal{A}, N))$ to $\mathcal{A}$.*

Running M_Ψ on a message m.

- *If M_Ψ was not initialized so far (i.e., if a message with ID $(\texttt{Ch}, \Psi)$ is received for the first time), initialize a new instance of the machine M as M_Ψ.*
- *Then / otherwise simulate M_Ψ on m (without its message ID), until it outputs a message m' and relay m', depending on its structure: to $\mathcal{A}$ (if $m' = (\texttt{Answer for}, \_, \_)$), or to $\texttt{Ch}$ (otherwise). In both cases, replace the message ID by $(\texttt{Ch}, \Psi)$.*

Theorem 2. *The Plug'n'Play adversary class is composable, i.e., all guarantees shown for 1 challenge scale linearly for multiple challenges.*

We refer to Section 8 for a proof.

Remark 1. *It is, theoretically, possible to compose adversary classes by nesting them into each other or by sequentially applying them. However, such a nested composition of adversary classes is then not necessarily single-challenge reducible in terms of Definition 9: The fact that answers from the challenger either are filtered twice (for adversary IDs and consequently for answers to the simulator) or input to a challenge-machine M in between the two filters can cause the simulation to fail. We could restrict the filter machine F to only apply to challenge messages, which would make such a composition possible, but this would also restrict the versatility of our* PAC *adversary class.*

We rather focus on an easy-to-use PAC *adversary class in which the machines themselves can be combinations of different interesting restrictions for the adversary.*

Example 7. *The following examples for adversary classes show the expressiveness of our* PAC *adversary class.*

(Interactive) profile adversaries: We set F to the identity machine (simply forwarding all inputs) and M to a profile machine that comprises of multiple different user profiles. Upon initialization with a message $(\texttt{Challenge}, m_1, m_2, \_)$ *the machine initializes two different profiles P_{m_1} and P_{m_2} in parallel. Consequently, M feeds all messages from the challenger into the profiles and outputs their answers as* `Input` *messages (if they are equal) or* `Challenge` *messages, if they are not equal.*

Fixed-user adversaries: We want the adversary to only select two predefined senders S_1, S_2 and/or two predefined recipients R_1, R_2 for its challenges. Consequently, we set F to the identity machine (simply forwarding all inputs) and define M as follows: M blocks all messages of the form $(\texttt{Input}, \_, \_)$ *and replaces all messages* $(\texttt{Challenge}, (S_x, m, R_x), (S_y, m, R_y), \_)$ *by* $(\texttt{Challenge}, (S_1, m, R_1), (S_2, m, R_2), \_)$*. This is a generalization of the* `MATor`*-adversary classes (Backes et al., 2014a; 2015b).*

Compromisation adversaries: We want to restrict the parties the adversary can compromise (in any arbitrary way, possibly depending on previous compromisation). We set M to the identity machine (simply forwarding all inputs) and F to the following machine: Whenever F receives a message $(\texttt{Protocol}, \text{compromise X})$*, F only forwards the message if $\mathcal{A}$ is allowed to compromise party X. This is a generalization of the budget adversary (Backes et al., 2015b).*

Example 8 (Metadata Adversary Classes)**.** *Metadata adversary classes hide the responses of the protocol from the adversary. This class of adversaries can be represented as a specific class of Plug'n'Play adversary classes. We solely restrict the challenge machine M to be of the form that they internally run an arbitrary machine M' and forward all messages from and to M' except for messages that M' intends to send to the adversary. Those messages are simply dropped.*

A metadata adversary class *is a Plug'n'Play adversary class such that the challenge machine M is of the following form:*

- *For every challenge tag ϕ, an instance M_ϕ has an arbitrary submachine M'.*
- *Upon being called with a message m call M' with m.*
- *Whenever M' wants to forward a message to the adversary, block this message. Forward all messages that M' wants to send to the challenger.*

Limitations. Our PAC adversary class is very versatile, but disallows adversary classes of the following types:

- Adversary classes that disallow or significantly restrict the adversary in sending messages of the form $(\texttt{Input}, \_, \_)$. Such adversary classes are rarely single-challenge reducible, and if they are, the respective proof has to be tailored to the adversary class.

- Adversary classes that transfer information between challenges in any way. Such adversary classes can, e.g., keep state in between different challenges or define consistent profiles that can be used for several circuits. Although such adversary classes may be of interest, they are typically not (in general) single-challenge reducible, as their key feature lies in challenges that depend upon each other.

7 Analyzing Tor Anonymity

The onion routing (OR) (Reed et al., 1998) network Tor (Tor Project) is the most successful anonymity technology to date: hundreds of thousands individuals all over the world use it today to protect their privacy over the Internet. Naturally, Tor is our first choice for applying our AnoA framework.

We start our discussion by briefly describing the Tor protocol (Dingledine et al., 2004) and its UC definition (Backes et al., 2012). We then formally prove (ε, δ)-IND-ANO for Tor's UC definition and quantify anonymity provided by the Tor network in terms of the anonymity properties defined in Section 5. Finally, we consider a selection of system-level attacks (e.g., traffic analysis) and adaptations (e.g., entry guards) for Tor, and analyze their effects on Tor's anonymity guarantees.

Note that, in the following, we present only a very basic analysis of the anonymity provided by Tor. A more sophisticated analysis that takes into account Tor's path selection algorithms and different adversarial capabilities can be found in follow up work (Backes et al., 2015b).

7.1 Tor – The OR Network

An OR network such as Tor (Dingledine et al., 2004) consists of a set of *OR nodes (or proxies)* that relay traffic, a large set of users and a directory service that maintains and provides cryptographic and routing information about the OR nodes. Users utilize the Tor network by selecting a sequence of OR nodes and creating a path, called a *circuit*, over this set. This circuit is then used to forward the users' traffic and obscure the users' relationship with their destinations. It is important that an OR node cannot determine the circuit nodes other than its immediate predecessor and successor. In the OR protocol, this is achieved by wrapping every message in multiple layers of symmetric-key encryption. Symmetric keys are agreed upon between each OR node in the circuit and the user during the circuit construction phase.

Tor was designed to guarantee anonymity against partially global adversaries, i.e., adversaries that do not only control some OR nodes but also a portion of the network. However, an accurate anonymity quantification is not possible without formally modeling the OR protocol and its adversary. In an earlier work, a formal UC definition (an ideal functionality $\mathcal{F}_{\text{OR}}$) for the OR network was presented, and a practical cryptographic instantiation was proposed (Backes et al., 2012). We employ this ideal functionality $\mathcal{F}_{\text{OR}}$ for instantiating the AnoA framework.

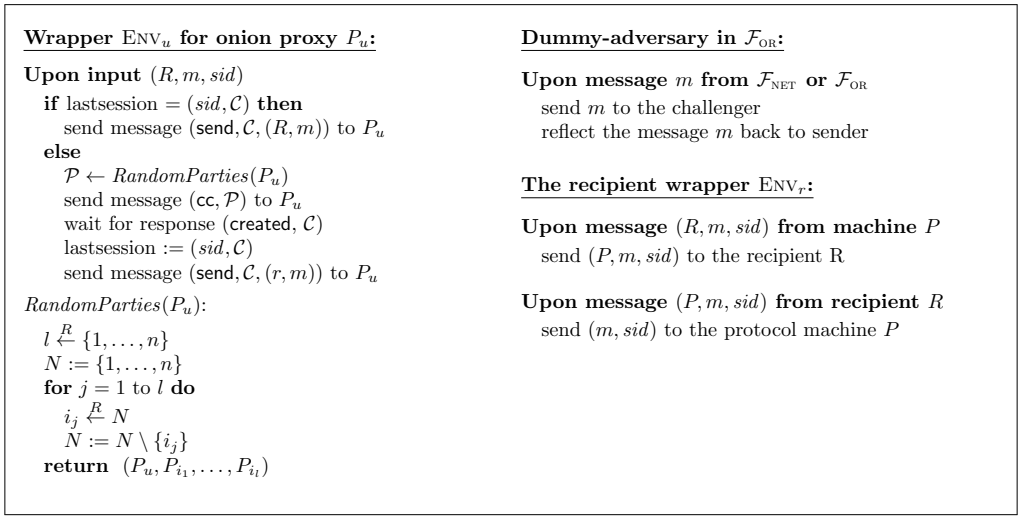

Figure 6: Extensions to $\mathcal{F}_{\text{OR}}$ for AnoA-Analysis.

Remark 2. *Our protocol model strengthens the adversarial control by allowing the adversary to receive all messages that users would receive. However, for correctly quantifying anonymity, we cannot forward messages related to challenges to the adversary. If we did, e.g., recipient anonymity would be trivially broken, as soon as the two recipients give different responses. In Section 8, we describe a generic solution to this trivial attack by introducing a class of meta-data attackers that do not get responses from the protocol.*

7.2 Excluding Website Fingerprint Attacks

As a low-latency AC network, Tor does not provide any protection against attacks that are based on traffic feature, such as direction changes or volume changes. In order to be able to quantify the recipient or relationship anonymity of Tor, we hence exclude website fingerprinting attacks by introducing a set of fingerprinting free webservers and introduce an adversary class that only forwards the traffic patterns to the adversary by replacing the content with constant-0 bit-strings.

Definition 11 (Direct connection protocol)**.** *Given a set of machines S, the direct connection protocol Di_S sends, upon a message (S, m, R, id) from the challenger, the message (m, id) to R on behalf of S over the network channel, however without leaking anything. Upon a response (r, id) from R, Di_S forwards (r, id) to the challenger.*

Definition 12 (Website Fingerprinting Free Servers)**.** *A pair P_0, P_1 of machine is* single-request website fingerprinting free *if for any pair of messages $m_0, m_1 \in \{0,1\}^*$ such that $|m_0| = |m_1|$ if P_i ($i \in \{0,1\}$) is sent the network message m_i then for the network response r_i of P_i, we have that $|r_0| = |r_1|$.*

A set S of machines is a set of single-request website fingerprinting free servers *if any pair of machines $P_0, P_1 \in S$ is* single-request website fingerprinting free*.*

Definition 13 (Traffic pattern leakage adversary class)**.** *A* traffic pattern leakage adversary class *is a Plug'n'Play adversary class such that the challenge machine M is of the following form:*

- *For every challenge tag ϕ, an instance M_ϕ has an arbitrary submachine M'.*
- *Upon being called with a message m call M' with m.*
- *Whenever M' outputs a message $m' = (\texttt{Answer for}, m, id)$, send $(\texttt{Answer for}, 0^{|m|}, id)$. If m' is of different form output m'.*

7.3 Anonymity Analysis

We start our Tor analysis with a brief overview of the $\mathcal{F}_{\text{OR}}$ functionality and refer the readers to Backes et al. (2012) for more details. $\mathcal{F}_{\text{OR}}$ presents the OR definition in the message-based state transitions form, and defines sub-machines for all OR nodes in the ideal functionality. These sub-machines share a memory space in the functionality for communicating with each other. $\mathcal{F}_{\text{OR}}$ assumes an adversary who might possibly control all communication links and destination servers, but cannot view or modify messages between uncompromised parties due to the presence of secure and authenticated channels between the parties. In $\mathcal{F}_{\text{OR}}$ these secure channels are realized by having each party store their messages in the shared memory, and create and send corresponding handles $\langle P, P_{nextt}, h\rangle$ through the network. Here, P and P_{next} are the sender and the recipient of a message respectively and h is a handle, or pointer, for the message in the shared memory. Only messages that are visible to compromised parties are forwarded to $\mathcal{A}$.

We consider a partially global, passive adversary for our analysis using AnoA, i.e., $\mathcal{A}$ decides on a subset of nodes before the execution, which are then compromised. The adversary $\mathcal{A}$ then only reads intercepted messages, but does not react to them.

Tor sets a time limit (of ten minutes) for each established circuit. However, the UC framework does not provide a notion of time. $\mathcal{F}_{\text{OR}}$ models such a time limit by only allowing a circuit C to transport at most a constant number (say ttl_C) of messages.

In the context of onion routing, we interpret an input message $r = (S, R, m, sid)$ as a message that is transmitted through the OR-network, where m is the message sent from the user S to the recipient R in the session that corresponds to the session with ID sid. The number of messages per session is bounded by ttl_C.

In order to make $\mathcal{F}_{\text{OR}}$ compatible with our `IND-ANO` definition, we require an additional wrapper functionality, which processes the input forwarded from the challenger `Ch`. This functionality is defined in Figure 6. Env_u receives any input from the challenger which had u as its user. If a circuit with session id sid already exists, the message is directly sent. Otherwise, Env_u initiates the circuit construction for a new session and sends the message contained through this circuit.

Messages intercepted by compromised nodes are sent to a network adversary $\mathcal{F}_{\text{NET}}$ described in Figure 6. $\mathcal{F}_{\text{NET}}$ forwards all intercepted messages to the challenger, who in turn forwards them to $\mathcal{A}$. Technically, the reflected messages should be of the form $\langle P, P_{\mathsf{next}}, h\rangle$ with the message handle h corresponding to the reflected message so that $\mathcal{F}_{\text{OR}}$ can continue with the computation. To simplify matters we just assume that everything works out correctly at this point.

Another exception are messages which are supposed to be sent to a recipient by the exit node, but would normally be sent to the adversary instead since $\mathcal{F}_{\text{OR}}$ does not contain the machines representing recipients. To alleviate this, we add another wrapper ENV_r which mediates the communication between the OR-protocol and the recipients $\mathcal{R}$ that are represented by `ppt` Turing machines. ENV_r internally simulates the execution of $\mathcal{F}_{\text{OR}}$ and the recipient machines and redirects all messages between those parties. Figure 6 defines the recipient wrapper ENV_r.

We show that the Tor analysis can be based on a *distinguishing event* $\mathcal{D}$, which has already been identified in the first onion routing anonymity analysis by Syverson et al. (2000, Fig. 1). The key observation is that the adversary can only learn about the sender or recipient of some message if he manages to compromise the entry- or exit-node of the circuit used to transmit this message. We define the distinguishing event $\mathcal{D}_\alpha$ for each of the anonymity notions defined in section 5.

(Session) Sender Anonymity (α_{sSA}). Let $\mathcal{D}_{\alpha_{sSA}}$ be the event that the entry-node of the challenge session is compromised by $\mathcal{A}$. This allows $\mathcal{A}$ to determine the sender of the challenge session and therefore break sender anonymity.

(Session) Recipient Anonymity (α_{sRA}). Let $\mathcal{D}_{\alpha_{sRA}}$ be the event that the exit-node of the challenge session is compromised by $\mathcal{A}$. This allows $\mathcal{A}$ to determine the recipient of the challenge session and therefore break recipient Anonymity.

(Session) Sender Unlinkability (α_{sUL}). Let $\mathcal{D}_{\alpha_{sUL}}$ be the event that $\mathcal{A}$ successfully compromises the entry nodes for both challenge sessions in the unlinkability game. This allows $\mathcal{A}$ to determine whether the sessions are linked or not, and hence break the unlinkability game.

(Session) Relationship Anonymity (α_{sREL}). Let $\mathcal{D}_{\alpha_{sREL}}$ be the event that $\mathcal{A}$ successfully compromises entry- and exit-node of the challenge session. This allows him to link both sender and recipient of the session.

We first prove Lemma 5. It captures anonymity provided by $\mathcal{F}_{\text{OR}}$ in case $\mathcal{D}$ does not happen. We then use Lemma 5 to prove (ε, δ)-`IND-ANO` for $\mathcal{F}_{\text{OR}}$ in general.

We introduce random strings ρ and $\rho_{\mathtt{Ch}}$ as additional input to the adversary and the challenger respectively. This allows us to handle them as deterministic machines and simplifies the proof for Lemma 5. Accordingly, all subsequent probabilities are taken over those random strings.

Lemma 5. *Let* $\rho, \rho_{\mathtt{Ch}} \xleftarrow{\mathcal{U}} \{0,1\}^{p(\eta)}$. *Given an adjacency function* $\alpha \in \{\alpha_{sSA}, \alpha_{sRA}, \alpha_{sUL}, \alpha_{sREL}\}$, *it holds that*

$$\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho);\rho,\rho_{\mathtt{Ch}} \xleftarrow{\mathcal{U}} \{0,1\}^{p(\eta)}\right]$$
$$=\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho);\rho,\rho_{\mathtt{Ch}} \xleftarrow{\mathcal{U}} \{0,1\}^{p(\eta)}\right].$$

We refer to Section 8 for the proof. With this result we obtain (ε,δ)- `IND-ANO` for $\mathcal{F}_{\text{OR}}$ by simple manipulation of equations.

Theorem 3. *$\mathcal{F}_{\text{OR}}$ is $(0,\delta)$ - `IND-ANO` for $\alpha \in \{\alpha_{sSA},\alpha_{sRA},\alpha_{sUL},\alpha_{sREL}\}$, i.e*

$$\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0\right] \leq \Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0\right] + \delta$$

with $\delta = \Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)]$.

We refer to Section 8 for a proof. Here δ is exactly the probability for the event $\mathcal{D}_\alpha$ that allows $\mathcal{A}$ to distinguish between both scenarios. Interestingly, we get the parameter value $\varepsilon = 0$. This implies that as long as $\mathcal{D}_\alpha$ does not happen, $\mathcal{F}_{\text{OR}}$ provides perfect anonymity for its users.

7.4 Anonymity Quantification

We evaluate the guarantees provided by Theorem 3 and consider further results we can derive from it for the special case of sender anonymity.

Distinguishing Events

We compute the probability of the distinguishing event $\mathcal{D}$ by counting the observations. For an OR network of n OR nodes such that k of those are compromised, probabilities associated with the various anonymity notions are as follows, where for recipient anonymity and relationship anonymity website fingerprinting sets of servers and the traffic pattern adversary class (see Section 7.2) are assumed:

Sender session Anonymity (α_{sSA}). $Pr[\mathcal{D}_{\alpha_{sSA}}] = 1 - \frac{\binom{n-1}{k}}{\binom{n}{k}} = \frac{k}{n}$.

Recipient session Anonymity (α_{sRA}). $Pr[\mathcal{D}_{\alpha_{sRA}}] = 1 - \frac{\binom{n-1}{k}}{\binom{n}{k}} = \frac{k}{n}$.

Sender session Unlinkability (α_{sUL}). $Pr[\mathcal{D}_{\alpha_{sUL}}] = \left(\frac{k}{n}\right)^2$.

Relationship session Anonymity (α_{sREL}). $Pr[\mathcal{D}_{\alpha_{sREL}}] = \frac{\binom{n-2}{k-2}}{\binom{n}{k}} = \frac{k(k-1)}{n(n-1)}$.

Note that the above analysis and the underlying model assume all OR nodes to be identical, and can perform all roles. Respecting OR node operators' legal boundaries, the real-world Tor network allows OR nodes to function in specific roles. To some extent, this simplifies $\mathcal{A}$'s task of identifying entry- or exit-nodes for circuits.

Multiple Challenges. In this section we considered the number of allowed challenges to be set to $n = 1$. However, scenarios in which the adversary can use more than one challenge can directly be derived by application of Theorem 1.

Adversary classes. The analysis of Tor conducted in this section does not explicitly mention adversary classes (cf. Definition 7). However, since a `ppt` adversary class $\mathrm{C}(\cdot)$ applied to a `ppt` adversary is again a `ppt` adversary, naturally, the analysis holds for arbitrary adversary classes.

Link Corruption. Link corruption is not explicitly covered in this example analysis. We refer to the full example analysis (Backes et al., 2013) for a description of how to encode links.

7.5 Traffic Analysis Attacks

Many of the known attacks on Tor nowadays depend on so called side-channel information, i.e. throughput and timing information an adversary might gather while watching traffic routed through the Tor network. Since the UC framework does not allow time-sensitive attacks, traffic analysis is outside of the scope of this example analysis. However, due to the strong adversary we deploy, we can still cover traffic analysis attacks such as traffic correlation by making suitable assumptions.

In a traffic correlation attack, the adversary observes traffic going out from the sender and into the receiver and tries to correlate them based on different features like volume, direction or inter-packet delay (O'Gorman and Blott, 2009; Wang et al., 2002). We model these attacks by over-approximating the adversary. We allow adversaries that only send a single challenge. If the adversary controls the links or nodes that are needed for a traffic correlation attack, i.e., the entry and exit node or the link from the user to the entry node and the link from the exit node to the server, then the adversary can correlate the traffic.

8 Conclusion & Future Work

In this paper we have presented AnoA, a generic and versatile framework for formally quantifying the anonymity provided by AC protocols. Together with the description of our framework we have presented novel, strong variants of well-studied anonymity notions, such as sender anonymity, sender unlinkability, recipient anonymity and relationship anonymity based on computational indistinguishability in the spirit of (computational) differential privacy. We have shown that our definitions of anonymity guarantees accurately model prominent notions in the literature.

Moreover, we have presented *adversary classes*, an intuitive concept of weakening the adversary in order to analyze more realistic attack scenarios. We have shown that for the anonymity notions defined in AnoA and for many adversary classes it suffices to

analyze anonymity in a one-challenge-only game. We have found and defined a property which we coin Single-challenge reducibility that an adversary class must fulfill for this result to hold and we have even defined an easy to use structure for adversary classes, called *Plug'n'Play adversary classes* (PAC) that describe a wide variety of interesting adversary classes and that all are single-challenge reducible.

We have shown how to apply AnoA to an AC protocol by conducting an example-analysis of the Tor network. We have validated the inherent imperfection of the current Tor even under idealistic assumptions, and we have given quantitative guarantees for the different notions of anonymity against passive adversaries that statically corrupt nodes. In succession to our analysis, Backes et al. utilized AnoA to perform an extended, more thorough analysis of Tor that takes into account Tor's actual path selection algorithm as well as its current network status (Backes et al., 2014a; 2015b).

So far, all analyses that utilize the AnoA framework analyze the Tor network and are restricted to passive adversaries. Consequently, the next steps include to analyze Tor's anonymity against more powerful (active) adversaries and to also apply AnoA to other AC protocols, such as Mixnets (Chaum, 1981) and the DISSENT system (Corrigan-Gibbs and Ford, 2010). Moreover, AnoA purposefully is oblivious to the semantic leakage of information within messages. We will investigate a generic way to combine AnoA's anonymity analysis of an AC protocol with semantical analyses of user behavior.

References

Andrés, E., Miguel, Palamidessi, C., Sokolova, A., and Van Rossum, P. (2011). "Information Hiding in Probabilistic Concurrent Systems." *Journal of Theoretical Computer Science (TCS)*, 412(28): 3072–3089.

Backes, M., Goldberg, I., Kate, A., and Mohammadi, E. (2012). "Provably Secure and Practical Onion Routing." In *Proc. 26st IEEE Symposium on Computer Security Foundations (CSF)*, 369–385.

Backes, M. and Jacobi, C. (2003). "Cryptographically Sound and Machine-Assisted Verification of Security Protocols." In *Proceedings of 20th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 675–686.

Backes, M., Kate, A., Manoharan, P., Meiser, S., and Mohammadi, E. (2013). "AnoA: A Framework for Analyzing Anonymous Communication Protocols." In *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, 163–178. IEEE.

Backes, M., Kate, A., Meiser, S., and Mohammadi, E. (2014a). "(Nothing else) MATor (s): Monitoring the Anonymity of Tor's Path Selection." In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 513–524. ACM.

Backes, M., Kate, A., Meiser, S., and Ruffing, T. (2015a). "Secrecy without Perfect Randomness: Cryptography with (Bounded) Weak Sources." In *Proceedings of the 13th International Conference on Applied Cryptography and Network Security, (ACNS'15)*.

Backes, M., Manoharan, P., and Mohammadi, E. (2014b). "TUC: Time-sensitive and Modular Analysis of Anonymous Communication." In *Proc. 27th IEEE Computer Security Foundations Symposium (CSF)*, 383–397. IEEE.

Backes, M., Meiser, S., and Slowik, M. (2015b). "Your Choice MATor (s)." *Proceedings on Privacy Enhancing Technologies*, 2016(2): 40–60.

Backes, M., Pfitzmann, B., and Waidner, M. (2007). "The Reactive Simulatability (RSIM) Framework for Asynchronous Systems." *Information and Computation*, 205(12): 1685–1720.

Bhargava, M. and Palamidessi, C. (2005). "Probabilistic Anonymity." In *CONCUR*, 171–185.

Cai, X., Zhang, X. C., Joshi, B., and Johnson, R. (2012). "Touching from a Distance: Website Fingerprinting Attacks and Defenses." In *Proc. 19th ACM Conference on Computer and Communication Security (CCS)*, 605–616.

Camenisch, J. and Lysyanskaya, A. (2005). "A Formal Treatment of Onion Routing." In *Advances in Cryptology — CRYPTO*, 169–187.

Canetti, R. (2013). "Universally Composable Security: A New Paradigm for Cryptographic Protocols." Cryptology ePrint Archive, Report 2000/067.

Chaum, D. (1981). "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." *Communications of the ACM*, 4(2): 84–88.

— (1988). "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability." *J. Cryptology*, 1(1): 65–75.

Corrigan-Gibbs, H. and Ford, B. (2010). "Dissent: Accountable Anonymous Group Messaging." In *Proc. 17th ACM Conference on Computer and Communication Security (CCS)*, 340–350.

Danezis, G. and Goldberg, I. (2009). "Sphinx: A Compact and Provably Secure Mix Format." In *Proc. 30th IEEE Symposium on Security and Privacy*, 269–282.

Díaz, C. (2006). "Anonymity Metrics Revisited." In *Anonymous Communication and its Applications*.

Díaz, C., Seys, S., Claessens, J., and Preneel, B. (2002). "Towards Measuring Anonymity." In *Proc. 2nd Workshop on Privacy Enhancing Technologies (PET)*, 54–68.

Dingledine, R., Mathewson, N., and Syverson, P. (2004). "Tor: The Second-Generation Onion Router." In *Proc. 13th USENIX Security Symposium (USENIX)*, 303–320.

Dwork, C. (2006). "Differential Privacy." In *ICALP (2)*, 1–12.

— (2008). "Differential Privacy: A Survey of Results." In *TAMC*, 1–19.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). "Calibrating Noise to Sensitivity in Private Data Analysis." In *Proc. 10th Theory of Cryptography Conference (TCC)*, 265–284.

Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. (2012). "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail." In *Proc. 33th IEEE Symposium on Security and Privacy*, 332–346.

Feigenbaum, J., Johnson, A., and Syverson, P. F. (2007a). "A Model of Onion Routing with Provable Anonymity." In *Proc. 11th Conference on Financial Cryptography and Data Security (FC)*, 57–71.

— (2007b). "Probabilistic Analysis of Onion Routing in a Black-Box Model." In *Proc. 6th ACM Workshop on Privacy in the Electronic Society (WPES)*, 1–10.

— (2012). "Probabilistic Analysis of Onion Routing in a Black-Box Model." *ACM Transactions on Information and System Security (TISSEC)*, 15(3): 14.

Gelernter, N. and Herzberg, A. (2013). "On the limits of provable anonymity." In *Proc. 12th ACM Workshop on Privacy in the Electronic Society (WPES)*, 225–236.

Gierlichs, B., Troncoso, C., Díaz, C., Preneel, B., and Verbauwhede, I. (2008). "Revisiting a Combinatorial Approach toward Measuring Anonymity." In *Proc. 7th ACM Workshop on Privacy in the Electronic Society (WPES)*, 111–116.

Halpern, J. Y. and O'Neill, K. R. (2005). "Anonymity and Information Hiding in Multiagent Systems." *Journal of Computer Security*, 13(3): 483–512.

Hevia, A. and Micciancio, D. (2008). "An Indistinguishability-Based Characterization of Anonymous Channels." In *Proc. 8th Privacy Enhancing Technologies Symposium (PETS)*, 24–43.

Hofheinz, D. and Shoup, V. (2013). "GNUC: A New Universal Composability Framework." *Journal of Cryptology*, 1–86.

Hughes, D. and Shmatikov, V. (2004). "Information Hiding, Anonymity and Privacy: a Modular Approach." *Journal of Computer Security*, 12(1): 3–36.

Kate, A. and Goldberg, I. (2010). "Using Sphinx to Improve Onion Routing Circuit Construction." In *Proc. 14th Conference on Financial Cryptography and Data Security (FC)*, 359–366.

Küsters, R. and Tuengerthal, M. (2013). "The IITM Model: a Simple and Expressive Model for Universal Composability." *IACR Cryptology ePrint Archive*, 2013: 25.

Mauw, S., Verschuren, J. H., and de Vink, E. P. (2004). "A Formalization of Anonymity and Onion Routing." In *Proc. 9th European Symposium on Research in Computer Security (ESORICS)*, 109–124.

O'Gorman, G. and Blott, S. (2009). "Improving Stream Correlation Attacks on Anonymous Networks." In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC)*, 2024–2028.

Panchenko, A., Niessen, L., Zinnen, A., and Engel, T. (2011). "Website Fingerprinting in Onion Routing Based Anonymization Networks." In *Proc. 10th ACM Workshop on Privacy in the Electronic Society (WPES)*, 103–114.

Pfitzmann, A. and Hansen, M. (2010). "A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management." http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. V0.34.

Reed, M. G., Syverson, P. F., and Goldschlag, D. M. (1998). "Anonymous Connections and Onion Routing." *IEEE J-SAC*, 16(4): 482–494.

Reiter, M. K. and Rubin, A. D. (1998). "Crowds: Anonymity for Web Transactions." *ACM Transactions on Information and System Security (TISSEC)*, 1(1): 66–92.

Serjantov, A. and Danezis, G. (2002). "Towards an Information Theoretic Metric for Anonymity." In *Proc. 2nd Workshop on Privacy Enhancing Technologies (PET)*, 41–53.

Shmatikov, V. (2004). "Probabilistic Analysis of an Anonymity System." *Journal of Computer Security*, 12(3-4): 355–377.

Shmatikov, V. and Wang, M.-H. (2006). "Measuring Relationship Anonymity in Mix Networks." In *Proc. 7th ACM Workshop on Privacy in the Electronic Society (WPES)*, 59–62.

Syverson, P., Tsudik, G., Reed, M., and Landwehr, C. (2000). "Towards an Analysis of Onion Routing Security." In *Proc. Workshop on Design Issues in Anonymity and Unobservability (WDIAU)*, 96–114.

Tor Project (....). "The Tor Project." https://www.torproject.org/. Accessed in May 2014.

Wang, X., Reeves, D. S., and Wu, S. F. (2002). "Inter-Packet Delay Based Correlation for Tracing Encrypted Connections through Stepping Stones." In *Proc. 7th European Symposium on Research in Computer Security (ESORICS)*, 244–263.

Wikström, D. (2004). "A Universally Composable Mix-Net." In *Proc. of the 1st Theory of Cryptography Conference (TCC)*, 317–335.

A. Proofs for Single-Challenge Reducibility

A.1 Proof for Theorem 1

Proof. We will show this theorem inductively. Assume that Π is $(i, i\cdot\varepsilon, e^{i\varepsilon}\cdot i\cdot\delta)$-IND-ANO. We show that Π is also $(i+1, (i+1)\cdot\varepsilon, e^{(i+1)\cdot\varepsilon}(i+1)\cdot\delta)$-IND-ANO.

Let $\mathcal{A}$ be an adversary that sends at most $i+1$ challenges. To do so, we construct several games:

- **Game:** G_0 is the normal game $Real^{(\mathrm{C}\leftrightarrow\mathcal{A})}(0, i+1)$ with up to $i+1$ challenges where $b = 0$.
- **Game:** G_1 is an intermediate game $Sim_{\mathtt{S}_{\mathtt{zreal}}}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(0, i+1)$. Here every message from $\mathcal{A}$ to $\mathrm{C}(\mathcal{A})$ (and otherwise) goes through the simulator $\mathtt{S}_{\mathtt{zreal}}$. However, this simulator does not need to simulate anything, as there are still up to $i+1$ challenges and $b = 0$.

  Claim: G_0 and G_1 are computationally indistinguishable.

  Proof: By item 3b Definition 9 the simulator $\mathtt{S}_{\mathtt{zreal}}$ exists and the games are indistinguishable.
- **Game:** G_2 is an intermediate (hybrid) game $Sim_{\mathtt{S}_{\mathtt{z}}}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(0, i+1)$ with $b = 0$ and fixed `Input` messages instead of the challenge with tag $i+1$ (so there are at most i challenges left). This is done by using the simulator $\mathtt{S}_{\mathtt{z}}$ for $\mathtt{z} = [(dontsimulate, \_), \ldots, (dontsimulate, \_), (simulate, 0)] \in \{0,1\}\, i+1$, i.e., the simulator simulates the $i+1$st challenge for $b = 0$.

  Claim: G_1 and G_2 are computationally indistinguishable.

Proof: By item 3 of Definition 9, we know that the simulator $\mathtt{S}_\mathtt{z}$ exists. Since the simulator $\mathtt{S}_\mathtt{z}$ from G_2 uses the correct bit $b_{i+1} = 0$ for the simulated challenge, Item 3c of Definition 9 implies that the games are indistinguishable.

- **Game:** G_3 is the intermediate (hybrid) game $Sim_{\mathtt{S}_\mathtt{z}}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(1, i+1)$ where the simulator stays $\mathtt{S}_\mathtt{z}$ but the challenger changes to $b = 1$.

  Claim: G_2 and G_3 are $(i\varepsilon, e^{i\varepsilon} i\delta)$-indistinguishable.

  Proof: The adversary $\mathtt{S}_\mathtt{z}(\mathcal{A})$ makes at most i queries with challenge tags in $\{1, \ldots, i\}$. From the reliability property of the adversary class (item 1 of Definition 9) we know that thus $\mathrm{C}(\mathtt{S}_\mathtt{z}(\mathcal{A}))$ uses at most i challenge tags in $\{1, \ldots, i\}$. The claim immediately follows from the induction hypothesis: Π is $(i, i\cdot\varepsilon, i\cdot\delta)$-`IND-ANO`.

- **Game:** G_4 is a game $Sim_{\mathtt{S}_{\mathtt{z}'}}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(1, i+1)$ where the simulator $\mathtt{S}_{\mathtt{z}'}$ with $\mathtt{z}' = [(simulate, 1), \ldots, (simulate, 1), (simulate, 0)]$ simulates all challenges from $\mathcal{A}$. For the challenge tags 1 to i, $\mathtt{S}_{\mathtt{z}'}$ simulates the challenges for $b_1 = \ldots = b_i = 1$, whereas for the tag $i+1$ it still simulates it for $b_{i+1} = 0$. The challenger uses $b = 1$.

  Claim: G_3 and G_4 are computationally indistinguishable.

  Proof: Since the simulator $\mathtt{S}_{\mathtt{z}'}$ from G_4 uses the correct bit $b_1 = \ldots = b_i = 1$ for the challenges that are not simulated in $\mathtt{S}_\mathtt{z}$, Item 3c of Definition 9 implies that the games are indistinguishable.

- **Game:** G_5 is the game $Sim_{\mathtt{S}_{\mathtt{z}'}}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(0, i+1)$ where we use the same simulator $\mathtt{S}_{\mathtt{z}'}$ but we have $b = 0$ again.

  Claim: G_4 and G_5 are computationally indistinguishable.

  Proof: Since there are no challenge messages (everything is simulated, as by item 3a $\mathtt{S}_{\mathtt{z}'}$ does not send any messages $(\mathtt{Challenge}, \_, \_, \Psi)$), changing the bit b of the challenger does not have any effect. Hence, the games are indistinguishable.

- **Game:** G_6 is the game $Sim_{\mathtt{S}_{\mathtt{z}''}}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(0, i+1)$ where we use the simulator $\mathtt{S}_{\mathtt{z}''}$ with $\mathtt{z}'' = [(simulate, 1), \ldots, (simulate, 1), (dontsimulate, \_)]$. In other words, we do not simulate the challenge for $i+1$ with $b_{i+1} = 0$, but we use the challenger again (also with $b = 0$).

  Claim: G_5 and G_6 are computationally indistinguishable.

  Proof: Since the simulator $\mathtt{S}_{\mathtt{z}'}$ from G_5 uses the correct bit $b_{i+1} = 0$ for the simulated challenge (which the simulator $\mathtt{S}_{\mathtt{z}''}$ does not simulate), Item 3c of Definition 9 implies that the games are indistinguishable.

- **Game:** G_7 is $Sim_{translator(\mathtt{S}_{\mathtt{z}''})}^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(0, i+1)$ where we build around the simulator $\mathtt{S}_{\mathtt{z}''}$ an interface $translator(\cdot)$ that translates the challenge tag from $i+1$ to 1 and vice versa in all messages $(\mathtt{Challenge}, \_, \_, \Psi)$ from $\mathtt{S}_{\mathtt{z}''}$ to $\mathrm{C}(\mathcal{A})$ and in all messages $(\mathtt{Answer\ for}, \_, \Psi)$ from $\mathrm{C}(\mathcal{A})$ to $\mathtt{S}_{\mathtt{z}''}$..

  Claim: G_6 and G_7 are information theoretically indistinguishable.

  Proof: Item 2 of Definition 9 requires that the renaming of challenge tags does not influence the behavior of $\mathrm{C}(\mathcal{A})$. It also does not influence the behavior of the

challenger (by definition) or the protocol (that never sees challenge tags). Thus, the games are indistinguishable.

- **Game:** G_8 is the game $Sim^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}_{translator(\mathtt{S}_{\mathtt{z}''})}(1, i+1)$ where the simulator is defined as in G_7 but $b = 1$.

  Claim: G_7 and G_8 are (ε, δ) indistinguishable.

  Proof: By assumption of the theorem, the protocol Π is $(1, \varepsilon, \delta)$-IND-ANO for $\mathrm{C}(\mathcal{A})$. Moreover, by definition of $\mathtt{z}''$ and by item 3a, the adversary $translator(\mathtt{S}_{\mathtt{z}''}(\mathcal{A}))$ only uses at most one challenge tag, namely the tag 1. From the reliability property of the adversary class (item 1 of Definition 9) we know that thus $\mathrm{C}(translator(\mathtt{S}_{\mathtt{z}''}(\mathcal{A})))$ uses only the challenge tag 1. Thus, G_7 and G_8 are (ε, δ) indistinguishable.

- **Game:** G_9 is $Sim^{(\mathrm{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}_{\mathtt{S}_{\mathtt{z}''}}(1, i+1)$ where we remove the translation interface again.

  Claim: G_8 and G_9 are information theoretically indistinguishable.

  Proof: As before, Item 2 of Definition 9 requires that the renaming of challenge tags does not influence the behavior of $\mathrm{C}(\mathcal{A})$. It also does not influence the behavior of the challenger (by definition) or the protocol (that never sees challenge tags). Thus, the games are indistinguishable.

- **Game:** G_{10} is the normal game $Real^{(\mathrm{C}\leftrightarrow\mathcal{A})}(1, i+1)$ where $b = 1$.

  Claim: G_9 and G_{10} are computationally indistinguishable.

  Proof: Since $\mathtt{S}_{\mathtt{z}''}$ uses the correct bit $b_1 = \ldots = b_i = 1$ for all simulations, we can replace it with $\mathtt{S}_{\mathtt{zreal}}$, that, in turn, is indistinguishable from $Real^{(\mathrm{C}\leftrightarrow\mathcal{A})}(1, i+1)$.

We slightly abuse notation in writing $\Pr[0 = \mathcal{A}(G_0)]$ for $\Pr[0 = \langle \mathrm{C}(\mathcal{A}(n)) | \mathtt{Ch}(\Pi, \alpha, n, 0)\rangle]$, $\Pr[0 = \mathcal{A}(G_1)]$ for $\Pr[0 = \langle \mathrm{C}(\mathtt{S}_{\mathtt{z}}(b, \mathcal{A}(n))) | \mathtt{Ch}(\Pi, \alpha, n, 0)\rangle]$, etc..

$$
\begin{aligned}
&\Pr\left[0 = \mathcal{A}(G_0)\right] \\
&\leq \Pr\left[0 = \mathcal{A}(G_1)\right] + \mu_1 \\
&\leq \Pr\left[0 = \mathcal{A}(G_2)\right] + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon}\Pr\left[0 = \mathcal{A}(G_3)\right] + e^{i\varepsilon} i\delta + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon}\Pr\left[0 = \mathcal{A}(G_4)\right] + e^{i\varepsilon}(\mu_3 + i\delta) + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon}\Pr\left[0 = \mathcal{A}(G_5)\right] + e^{i\varepsilon}(\mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon}\Pr\left[0 = \mathcal{A}(G_6)\right] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&= e^{i\varepsilon}\Pr\left[0 = \mathcal{A}(G_7)\right] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon}(e^{\varepsilon}\Pr\left[0 = \mathcal{A}(G_8)\right] + \delta) + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&= e^{(i+1)\varepsilon}\Pr\left[0 = \mathcal{A}(G_8)\right] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + (i+1)\delta) + \mu_2 + \mu_1 \\
&= e^{(i+1)\varepsilon}\Pr\left[0 = \mathcal{A}(G_9)\right] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + (i+1)\delta) + \mu_2 + \mu_1 \\
&\leq e^{(i+1)\varepsilon}\Pr\left[0 = \mathcal{A}(G_{10})\right] + e^{(i+1)\varepsilon}\mu_6 + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + (i+1)\delta) + \mu_2 + \mu_1 \\
&\leq e^{(i+1)\varepsilon}\Pr\left[0 = \mathcal{A}(G_{10})\right] + e^{(i+1)\varepsilon}(i+1)\delta.
\end{aligned}
$$

□

A.2 Proof for Theorem 2

Proof. We start the proof by showing that any Plug'n'Play adversary class allows for single-challenge reducibility.

- **Alpha-Renaming:** The Plug'n'Play adversary class $\mathrm{PAC}_{M,F}$ uses challenge tags Ψ exactly for one purpose: for handling one instance of a machine M per challenge. The behavior of the machines F and M is not affected by Ψ (simply because they never see them). Thus, for any permutation on the challenge tags, the adversary class will have exactly the same behavior.
- **Reliability:** Note that the challenger never initializes challenges, i.e., will never send any message containing $(\texttt{Ch}, \Psi)$ before receiving a message containing $(\texttt{Ch}, \Psi)$. By construction, $\mathrm{PAC}_{M,F}$ only sends messages with ID $(\texttt{Ch}, \Psi)$ when M_Ψ outputs them. However, $\mathrm{PAC}_{M,F}$ invokes M_Ψ only if it (before) receives a message with ID $(\texttt{Ch}, \Psi)$. Since the challenger cannot send a message with this ID before receiving a message with this ID, $\mathrm{PAC}_{M,F}$ only initializes and invokes M_Ψ if the adversary has send a message with ID $(\texttt{Ch}, \Psi)$ before. Thus, $\mathrm{PAC}_{M,F}$ is reliable.
- **Simulatability:** We construct the following *general simulator* `S`:
  - `S` initialized a set of known IDs $\mathcal{I} := \emptyset$.
  - Whenever `S` receives any message $m = (\ldots, ID)$, where $ID = (\texttt{Ch}, \Psi)$ with $z_\Psi = \texttt{don't simulate}$, `S` forwards the message.

- Whenever $\mathtt{S}$ receives any message $m = (\ldots, ID)$, where $ID = (\mathcal{A}, x)$, s.t. $\forall z.(z, x) \notin \mathcal{I}$, $\mathtt{S}$ forwards the message.
- Whenever $\mathtt{S}$ receives any message $m = (\ldots, ID)$, where $ID = (\mathtt{Ch}, \Psi)$ with $z_\Psi = \mathtt{simulate}$ (for the bit b_Ψ), behave as follows:
  - If M_Ψ was not initialized so far (i.e., if a message with ID $(\mathtt{Ch}, \Psi)$ is received for the first time), initialize a new instance of the machine M as M_Ψ. Moreover, draw a new nonce N and add (Ψ, N) to $\mathcal{I}$.
  - Then / otherwise simulate M_Ψ on m (without its message ID), until it outputs a message m' and relay m', depending on its structure:
    - If $m' = (\mathtt{Answer\ for}, \_, \_)$, relay m' to $\mathcal{A}$, but replace the ID with $(\mathtt{Ch}, \Psi)$.
    - If $m' = (\mathtt{Input}, \_, \_)$, relay m' to $\mathtt{Ch}$, but replace the ID with $(\mathcal{A}, N)$.
    - If $m' = (\mathtt{Challenge}, r_0, r_1, \_)$, compute $r^* := \alpha(r_0, r_1, b_\Psi)$ and send $(\mathtt{Input}, r^*, (\mathcal{A}, N))$ to $\mathtt{Ch}$.
- Whenever $\mathtt{S}$ receives any message $m = (\ldots, ID)$, where $ID = (\mathcal{A}, x)$, s.t. $\exists z.(z, x) \in \mathcal{I}$, $\mathtt{S}$ runs M_Ψ and handles its output as above.

We consequently show that the general simulator $\mathtt{S}$ satisfies all conditions from Definition 9, Item 3.

- **Item 3a:** Let $z \in \{0,1\}^{2n}$ be any simulator index and $i \in \{1, \ldots, n\}$, s.t., $z_i = simulate$. Then by construction $\mathtt{S}_z$ intercepts all messages with ID $(\mathtt{Ch}, i)$ and feeds them into a simulated machine M_i. Whenever this machine outputs a message $(\mathtt{Challenge}, \_, \_, \_)$, $\mathtt{S}$ replaces it by $(\mathtt{Input}, r^*, (\mathcal{A}, N))$ for some nonce N, where $r^* := \alpha(r_0, r_1, b_\Psi)$. Furthermore, $\mathtt{S}$ never introduces messages of the type $(\mathtt{Challenge}, \_, \_, (\mathtt{Ch}, j))$ for any j: it only forwards messages of this form if they are sent by $\mathcal{A}$ and if $z_j = \mathtt{don't\ simulate}$. Thus, $\mathtt{S}$ never sends a message $(\mathtt{Challenge}, \_, \_, (\mathtt{Ch}, i))$.
- **Item 3b:** The simulator $\mathtt{S}_{\mathtt{zreal}}$ forwards all messages it receives. It never intercepts messages with ID $(\mathtt{Ch}, i)$ for any i and consequently, the set $\mathcal{I}$ is empty. Thus, it never intercepts messages with an ID $(\mathcal{A}, x)$ for any x either. Thus, $\mathtt{S}_{\mathtt{zreal}}$
- **Item 3c:** We now show that our simulator correctly simulates challenges (if the bit within its index is correct). Let n be the number of allowed challenges, let b be the bit of the challenger and let $z, z' \in \{0,1\}^{2n}$ such that

  $$\forall i \in \{1, \ldots, n\} \text{ s.t. } z_i \neq z'_i.(z_i = simulate \Rightarrow b_i = b) \wedge (z'_i = simulate \Rightarrow b'_i = b).$$

  We now show that the gamed $Sim_{\mathtt{S}_z}^{(\mathrm{C} \leftrightarrow \mathtt{S} \leftrightarrow \mathcal{A})}(b, n)$ and $Sim_{\mathtt{S}_{z'}}^{(\mathrm{C} \leftrightarrow \mathtt{S} \leftrightarrow \mathcal{A})}(b, n)$ (as in Definition 9) are indistinguishable.

  We begin with noticing that both our generic simulator and the adversary class do not share state between different ID's: Their behavior on messages $(\_, \mathtt{ID})$ does not in any way depend on any communication or computation on messages $(\_, \mathtt{ID}')$ for any other $\mathtt{ID}'$. Consequently, we can analyze their behavior for each $\mathtt{ID}$ separately.

We furthermore assume that there are indexes $i \in \{1, \ldots, n\}$ s.t. $z_i \neq z'_i$. Let i be such an index and let us assume w.l.o.g., that $z_i = \mathit{simulate}$. The difference in behavior covers exactly the messages with $\mathtt{ID} = (\mathtt{Ch}, i)$ by $\mathcal{A}$.[3] To simplify the proof we can thus assume that there is exactly one difference (for index i).

Whenever a message with $\mathtt{ID} = (\mathtt{Ch}, i)$ is sent by $\mathcal{A}$ to $\mathtt{S}_{z'}$, the simulator simply forwards the message to $\mathrm{PAC}_{M,F}$, which, in turn, feeds it to the machine M_i. Analogously, $\mathtt{S}_z$ initializes a machine M_i and feeds it all such messages. Whenever in the game with $\mathtt{S}_{z'}$ the challenger replies to the adversary with a message $(\mathtt{Answer\ for}, x, (\mathtt{Ch}, i))$, $\mathrm{PAC}_{M,F}$ also feeds it to M_i. If these messages instead (as in the game with $\mathtt{S}_z$) have the ID $(\mathcal{A}, N)$ for any nonce N, then $\mathrm{PAC}_{M,F}$ simply forwards them and $\mathtt{S}_z$ feeds them into its machine M_i.

We proceed by an inductive proof over all messages sent to and received by M_i.

The first message must originate from $\mathcal{A}$, as the challenger never initializes challenges (i.e., it never sends messages $(\mathtt{Ch}, i)$ without receiving such messages before). Before $\mathcal{A}$ sends such a message, the two games $Sim_{\mathtt{S}_z}^{(\mathtt{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(b, n)$ and $Sim_{\mathtt{S}_{z'}}^{(\mathtt{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(b, n)$ are information theoretically indistinguishable (the simulators do not behave differently). For this first message m, $\mathtt{S}_z$ now creates a fresh nonce N for translating the challenge ID and then initializes the machine M_i with m. Analogously, $\mathtt{S}_{z'}$ simply forwards the message to $\mathrm{PAC}_{M,F}$ that initializes M_i with m. Both machines will behave indistinguishably, as they are initialized with the same message. The games only differ if M_i outputs a message $(\mathtt{Challenge}, r_0, r_1, \_)$. In $Sim_{\mathtt{S}_{z'}}^{(\mathtt{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(b, n)$ this message is sent to $\mathtt{Ch}$ which, since this is a challenge message, applies the anonymity function α to it (using challenge bit b) and forwards the resulting message to the protocol. In $Sim_{\mathtt{S}_z}^{(\mathtt{C}\leftrightarrow\mathtt{S}\leftrightarrow\mathcal{A})}(b, n)$, the simulator applies α to it (using the same challenge bit b) and forwards the resulting message as $(\mathtt{Input}, r^*, (\mathcal{A}, N)$ to $\mathrm{PAC}_{M,F}$, which forwards the message to $\mathtt{Ch}$, which, since this is an input message, directly forwards it to the protocol.[4]

Let us assume that at any point further on in the game(s), where the machine(s) M_i behaved indistinguishably and (so far) the challenger(s) also behaved indistinguishably. Then, by the same argument as above, the machine M_i will again behave indistinguishably (or it entails a distinguisher for the aforementioned indistinguishability). Again, the games only are structurally different if M_i outputs a message $(\mathtt{Challenge}, r_0, r_1, \_)$, but this message will be transformed to a message r^* for Π that is the same for both games.

[3] Technically, $\mathtt{S}_z$ translates the communication regarding this challenge ID to $(\mathcal{A}, N)$ when communicating to the challenger, for a random nonce N. The messages with $\mathtt{ID} = (\mathcal{A}, N)$ are of course also affected.

[4] Technically, the challenger handles sessions in both cases, once for ID $(\mathtt{Ch}, i)$, once for ID $(\mathcal{A}, N)$. Its behavior may only differ if there was a message with ID $(\mathcal{A}, N)$ before, or afterwards sent by the real adversary $\mathcal{A}$, but this occurs with negligible probability only.

This concludes the proof. □

B Leveraging UC Realizability

Our adversary model in AnoA is strong enough to capture well-known simulation-based composability frameworks (e.g., UC (Canetti, 2013), IITM (Küsters and Tuengerthal, 2013) or RSIM (Backes et al., 2007)). In Section 7 we apply AnoA to a model in the simulation-based universal composability (UC) framework. In this section, we briefly introduce the UC framework and then prove that Differential Privacy (and therefore also `IND-ANO`) are preserved under realization. This preservation allows for an elegant crypto-free anonymity proof for cryptographic AC protocols with AnoA.

The UC framework allows for a modular analysis of security protocols. In the framework, the security of a protocol is defined by comparing it with a setting in which all parties have a direct and private connection to a trusted machine that provides the desired functionality. As an example consider an authenticated channel between two parties Alice and Bob. In the real world Alice calls a protocol that signs the message m to be communicated. She then sends the signed message over the network and Bob verifies the signature. In the setting with a trusted machine T, however, we do not need any cryptographic primitives: Alice sends the message m directly to T. T in turn sends m to Bob, who trusts T and can be sure that the message is authentic. The trusted machine T is called the *ideal functionality*.

Security in the UC framework is defined as follows: A real protocol is secure if an execution of this real protocol is indistinguishable from an execution of the corresponding ideal functionality. Here, indistinguishability is defined in terms binary random variables which represent the output of the probabilistic real protocol and ideal functionality.

Definition 14 (Indistinguishability (Canetti))**.** *Two binary distribution ensembles X and Y are indistinguishable, denoted $X \approx Y$ if for every $c \in \mathbf{N}$ there is a $\eta_0 \in \mathbf{N}$ such that for all $\eta > \eta_0$ and all x we have that*

$$|Pr[X(\eta, x)] = 1 - Pr[Y(\eta, x)] = 1| < \delta' = \eta^{-c}$$

The Real World. For the process in the real world we introduce the random variable $Real_{\Pi,A,D}(\eta, x)$ which captures the interaction of a protocol Π with an adversary $\mathcal{A}$, observed by a distinguisher D.$Real_{\Pi,A,D}$ will denote the ensemble of all those distributions. Note that as we try to argue about `IND-ANO` , our input x will be a tuple of inputs (x_0, x_1).

The Ideal World. Similarly, we introduce the random variable $Ideal_{\mathcal{F},S,D}(\eta, x)$ which captures the interaction of an ideal functionality $\mathcal{F}$, a simulator S and the distinguisher. $Ideal_{\mathcal{F},S,D}$ will again denote the ensemble of such random variables.

If the execution of a real protocol is indistinguishable from the execution of the corresponding ideal functionality, we say that the protocol UC-realizes the ideal functionality.

Definition 15 (Realization in UC)**.** *A protocol* Π *UC-realizes an ideal functionality* F *if for every* ppt *adversary* $\mathcal{A}$ *of* Π *there exists a* ppt *simulator* S *such that for every* ppt *distinguisher* D *it holds that*

$$Real_{\Pi,A,D} \approx Ideal_{\mathcal{F},S,D}$$

B.1 Preservation of Differential Privacy

UC-realization does not only allow us to prove the security of the real protocol given a trivially secure ideal functionality, but also allows to lift security guarantees proven for the realized protocol to the realizing protocol: given the realization of a (ϵ, δ)-differentially-private ideal functionality by a protocol Π, we get differential privacy for Π as well. This result is motivated by the ideas presented in the result of integrity property conservation by simulation-based indistinguishability shown by Backes and Jacobi (2003, Thm. 1).

Theorem 4. *If* Π *UC-realizes an* (ϵ, δ)*-dp functionality* $\mathcal{F}$ *then* Π *is* (ϵ, Δ)*-dp with* $\Delta = \delta + \delta'$ *for some negligible value* δ'*.*

Proof. Given an (ϵ, δ)-dp functionality $\mathcal{F}$, assume Π UC-realizes $\mathcal{F}$, but Π is not (ϵ, Δ)-dp, i.e. there exist an adversary $\mathcal{A}$ and two inputs x_0 and x_1 s.t.

$$Pr[A(\Pi(x_1)) = 1] > e^{\epsilon} Pr[A(\Pi(x_0)) = 1] + \Delta$$

such that $\Delta \geq \delta + \delta'$ for a non negligible value δ'. We construct the following ppt distinguisher D that uses $\mathcal{A}$ in order to separate Π from $\mathcal{F}$:

1. choose $b \xleftarrow{R} \{0, 1\}$ uniformly at random
2. send x_b through the network
3. depending on the output b^* of the adversary:
   (a) if the adversary returns $b^* = b$, decide that you observed (Π, A) and output 1
   (b) otherwise decide that you observed $(\mathcal{F}, S)$ and output 0.

We now bound the probabilities $\Pr[Real_{\Pi,A,D}(\eta, (x_0, x_1)) = 1]$ and $\Pr[Ideal_{\mathcal{F},S,D}(\eta, (x_0, x_1)) = 1]$ as required for the lemma. We will use the expressions $A^b_{Real_{\Pi,D}}$ and $S^b_{Ideal_{\mathcal{F},D}}$ to denote the output of the adversary and simulator respectively during the execution after the distinguisher decided on a specific $b \in \{0, 1\}$. Using the assumption that Π is not

(ϵ, Δ)-differentially private, the first expression computes to

$$
\begin{aligned}
&\Pr[Real_{\Pi,A,D}(\eta,(x_0,x_1))=1]\\
=&\Pr[A^b_{Real_{\Pi,D}}=b]\\
=&\Pr[A^1_{Real_{\Pi,D}}=1]\cdot\Pr[\text{D chooses } x_1]+\Pr[A^0_{Real_{\Pi,D}}=0]\cdot\Pr[\text{D chooses } x_0]\\
=&\Pr[A(\Pi(x_1))=1]\cdot\Pr[b=1|b\xleftarrow{R}\{0,1\}]+\Pr[A(\Pi(x_0))=0]\cdot\Pr[b=0|b\xleftarrow{R}\{0,1\}]\\
=&\frac{1}{2}\left(\Pr[A(\Pi(x_1))=1]+\Pr[A(\Pi(x_0))=0]\right)\\
>&\frac{1}{2}\left(\Pr[A(\Pi(x_0))=1]e^\epsilon+\Delta+\Pr[A(\Pi(x_0))=0]\right)\\
=&\frac{1}{2}\left(\Pr[A(\Pi(x_0))=1]e^\epsilon+\Delta+1-\Pr[A(\Pi(x_0))=1]\right)\\
=&\frac{1}{2}\left((e^\epsilon-1)\Pr[A(\Pi(x_0))=1]+\Delta+1\right)\\
\geq&\frac{1}{2}\left(1+\Delta\right).
\end{aligned}
\tag{1}
$$

Using the (ϵ, δ)-differential privacy of $\mathcal{F}$, the second expression can be bound as follows

$$
\begin{aligned}
&\Pr[Ideal_{\mathcal{F},S,D}(\eta,(x_0,x_1))=1]\\
=&\Pr[S^b_{Ideal_{\mathcal{F},D}}=b]\\
=&\Pr[S^1_{Ideal_{\mathcal{F},D}}=1]\cdot\Pr[\text{D chooses } x_1]+\Pr[S^0_{Ideal_{\mathcal{F},D}}=0]\cdot\Pr[\text{D chooses } x_0]\\
=&\Pr[S(\mathcal{F}(x_1))=1]\cdot\Pr[b=1|b\xleftarrow{R}\{0,1\}]+\Pr[S(\mathcal{F}(x_0)))=0]\cdot\Pr[b=0|b\xleftarrow{R}\{0,1\}]\\
=&\frac{1}{2}\left(\Pr[S(\mathcal{F}(x_1))=1]+\Pr[S(\mathcal{F}(x_0))=0]\right)\\
\leq&\frac{1}{2}\left(\Pr[S(\mathcal{F}(x_0))=1]e^\epsilon+\delta+\Pr[S(\mathcal{F}(x_0))=0]\right)\\
=&\frac{1}{2}\left((1-\Pr[S(\mathcal{F}(x_0))=0])e^\epsilon+\delta+\Pr[S(\mathcal{F}(x_0))=0]\right)\\
=&\frac{1}{2}\left(e^\epsilon+(1-e^\epsilon)\Pr[S(\mathcal{F}(x_0))=0]+\delta\right)\\
\leq&\frac{1}{2}\left(e^\epsilon+1-e^\epsilon+\delta\right)\\
=&\frac{1}{2}\left(1+\delta\right).
\end{aligned}
\tag{2}
$$

Putting Equations 1 and 2 together, we then get

$$\Pr[Real_{\Pi,A,D}(\eta,(x_0,x_1))=1]-\Pr[Ideal_{\mathcal{F},S,D}(\eta,(x_0,x_1))=1]>\frac{1}{2}(\Delta-\delta)=\frac{1}{2}\delta'$$

for a non negligible value $\frac{\delta'}{2}$. Hence D is a `ppt` machine distinguishing (A,Π) from $(S,\mathcal{F})$ with more than negligible probability, contradicting the UC-realization of $\mathcal{F}$ by Π (cf

Definition 15). Therefore our initial assumption is wrong and Π is (ϵ, Δ)-differentially private. □

In the same vein as above, we can also show that IND-ANO is preserved by UC realization. As a consequence of this result, it suffices to apply AnoA to ideal functionalities: transferring the results to the real protocol weakens the anonymity guarantees only by a negligible amount.

Corollary 1. *Let* Π *be* (ϵ, δ)-IND-ANO *and* Π *be a protocol. If* Π *UC-realizes* Π *then* Π *is* (ϵ, Δ)-IND-ANO *with* $\Delta = \delta + \delta'$ *for some negligible value* δ'.

The above result, in combination with an ideal functionality for an AC protocol, is useful for analyzing the AC protocol with respect to our strong anonymity definitions. In the next section, we exemplify the approach by using an ideal functionality for Tor (Backes et al., 2012), showing that the anonymity analysis of Tor boils down to a purely combinatorial analysis.

C Proofs for the Example Analysis

C.1 Proof for Lemma 5

Proof. We fix the random string $\rho_{\mathtt{Ch}}$. This in turn fixes the circuits through which $\mathcal{F}_{\text{OR}}$ sends each message. As circuits are drawn independently from the messages transmitted, $\mathcal{F}_{\text{OR}}$ draws the same set of circuits to transmit either of the challenge inputs. The adversary $\mathcal{A}$ only observes messages of the form $m = (h, P_1, P_2[, P_3, \ldots, P_k, m'])$ that contain the following information:

a) which party P in $\mathcal{F}_{\text{OR}}$ the message m comes from,

b) which party P' in $\mathcal{F}_{\text{OR}}$ the message m is sent to,

c) which circuit was used by the circuit-ID (cid),

d) if the following nodes are compromised: $P_3, \ldots, P_k$

e) if the exit node node or the link from the exit node to the server is compromised: the message that is sent.

Observe that the message m' is the same in both scenarios and the handles h are freshly chosen and, hence, do not leak any information about the path or the sender.

For $\alpha \in \{\alpha_{sSA}, \alpha_{sUL}\}$, we know that by $\neg\mathcal{D}_\alpha$ the adversary does not compromise the entry node, or the link from the user to the entry node. Let $R \subseteq \{0,1\}^{p(\eta)}$ be the subset of all random strings ρ, for which $\langle \mathcal{A}(\rho) | \mathtt{Ch}(\mathcal{F}_{\text{OR}}, \alpha, 1, 0, \rho_{\mathtt{Ch}}) \rangle = 0$. As $\langle \mathcal{A}(\rho) | \mathtt{Ch}(\mathcal{F}_{\text{OR}}, \alpha, 1, b, \rho_{\mathtt{Ch}}) \rangle$ behaves deterministically, we know that exactly for every $\rho \in R$, $\langle \mathcal{A}(\rho) | \mathtt{Ch}(\mathcal{F}_{\text{OR}}, \alpha, 1, 1, \rho_{\mathtt{Ch}}) \rangle = 0$, as both $\mathtt{Ch}(\mathcal{F}_{\text{OR}}, \alpha, 1, 0, \rho_{\mathtt{Ch}})$ and $\mathtt{Ch}(\mathcal{F}_{\text{OR}}, \alpha, 1, 1, \rho_{\mathtt{Ch}})$ forward the same messages to $\mathcal{A}$.

If $\alpha = \alpha_{sREL}$, $\mathcal{A}$ might learn partial information by compromising either entry- or exit-node. But this only allows him to reduce the set of possible input tables to two, each of which could have been selected by only one of the challengers. By the same argument as above, if we fix ρ, $\mathcal{A}$ returns the same value, regardless of which challenger he interacts with. Hence $\mathcal{A}$ does not learn about the challenger's decision, and we get for any random string $\rho_{\mathtt{Ch}}$

$$\begin{aligned} &\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right] \\ =&\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right]. \end{aligned} \tag{3}$$

As the probabilities are the same for any random string $\rho_{\mathtt{Ch}}$, we then get

$$\begin{aligned} &\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)\right] \\ =&\frac{1}{2^{p(\eta)}}\cdot\sum_{\rho_{\mathtt{Ch}}\in\{0,1\}^{p(\eta)}}\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right] \\ =&\frac{1}{2^{p(\eta)}}\cdot\sum_{\rho_{\mathtt{Ch}}\in\{0,1\}^{p(\eta)}}\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right] \\ =&\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)\right]. \end{aligned}$$

□

C.2 Proof for Theorem 3

Proof.

$$\begin{aligned} &\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0\right] \\ =&\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)\right]\cdot\Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho))] \\ &+\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)\right]\cdot\Pr[\neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] \\ =&\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,0,\rho_{\mathtt{Ch}})\rangle = 0 \mid \mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right]\cdot\Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] \\ &+\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right]\cdot\Pr[\neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] \\ \leq&\Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] + \Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right]\cdot\Pr[\neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] \\ \leq&\Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] + \Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right]\cdot\Pr[\neg\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] \\ &+\Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0 \mid \mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho),\rho_{\mathtt{Ch}}\right]\cdot\Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] \\ =&\Pr[\mathcal{D}_\alpha(\rho_{\mathtt{Ch}},\rho)] + \Pr\left[\langle\mathcal{A}(\rho)|\mathtt{Ch}(\mathcal{F}_{\text{OR}},\alpha,1,1,\rho_{\mathtt{Ch}})\rangle = 0\right]. \end{aligned}$$

□