Another Look at Anonymous Communication
Security and Modular Constructions

Russell W.F. Lai, Henry K.F. Cheung, Sherman S.M. Chow*, and Anthony Man-Cho So

The Chinese University of Hong Kong, Sha Tin, N.T., Hong Kong

Abstract. Anonymous communication is desirable for personal, financial, and political reasons. Despite the abundance of frameworks and constructions, the anonymity definitions are usually either not well-defined or too complicated to use. In between two extremes are ad-hoc definitions, but they only work for specific protocols and sometimes only provide weakened anonymity guarantees. It is fair to say that constructing and analyzing anonymous communication protocols is not an easy task.
This paper addresses this situation from the perspectives of syntax, security definition, and construction. We propose simple yet expressive syntax and security definition for anonymous communication. Our syntax covers protocols with different operational characteristics. We give a hierarchy of anonymity definitions, starting from the strongest possible one, to several relaxations. We also propose a modular construction of anonymous communication protocol, specifically, from any key-private public-key encryption scheme, and a new primitive we formulate which is called oblivious forwarding protocols.
We then give two constructions of oblivious forwarding protocols. Our first design is a generic construction from any random walk algorithm over graphs, while the second design is optimized for probability of successful delivery, with experimental validation for our optimization. Our protocols provide anonymity even when the adversary can observe and control all traffic in the network and corrupt all but two nodes, in contrast to some efficient yet not-so-anonymous communication protocols. We hope this work can serve as an easier way to design and analyze efficient anonymous communication protocols in the future.

1 Introduction

Since the seminal work of Chaum [11], the notion of anonymous communication has been extensively studied in the past decades. The goal of anonymous communication is to hide the correspondence between senders and receivers of messages. In a stricter sense, the identities of the senders and/or receivers may also need to be hidden. There are plentiful reasons for having anonymous communication, such as to act against censorship and mass surveillance, to protect the privacy of personal preferences, and to express minority opinions. The use of anonymous communication has become increasingly popular among the general public, as indicated by the success of the Tor network [17].

1.1 Anonymity against a Global Adversary

Very often, research on anonymous communication protocols focuses on achieving low latency, while the anonymity guarantee is not well defined. Pfitzmann and Hansen [27] consolidated informally a collection of terminologies (*e.g.*, unlinkability, anonymity, unobservability) which are commonly used in the literature. Hevia and Micciancio [20] formally gave indistinguishability-based definitions of many of these terminologies, and showed that unobservability is the strongest notion against passive eavesdroppers, yet all the definitions are actually equivalent under efficient transformations. Gelernter and Herzberg [19] extended the work of Hevia and Micciancio [20] to the setting with adaptive adversaries including malicious receivers. In particular, sender anonymity against malicious

* Corresponding author

receivers is considered the strongest anonymity possible in this setting. Unfortunately, not many of the recent works used these formal definitions. These definitions can be too complicated, as admitted by Gelernter and Herzberg [19], or not that well-known to the practical community. It is desirable to have a more accessible security definition, as *simple* as the indistinguishability definition (IND-CPA/CCA) for public-key encryption, yet *expressive* enough to capture the security properties desired by anonymous communication protocols.

A particular class of anonymous communication systems aims to provide provable anonymity (under corresponding ad-hoc definitions) with the presence of adversaries which globally observe all traffic of the network. Perhaps the most basic protocol within this class is the buses protocol [7], which circulates a large array of ciphertexts (the bus) along a fixed route covering all nodes in the network. The reduced-seats buses protocol [21] and the taxis protocol [22] have improved efficiency upon the buses protocol by reducing the size of the ciphertext carrier. At its extreme, Young and Yung [32] recently proposed the *Drunk Motorcyclist* (DM) protocol where each ciphertext carrier (the motorcycle) only carries a single ciphertext. The ciphertext only travels to a random neighboring node upon arriving each node, hence the name Drunk Motorcyclist. Young and Yung [32] also fixed a flaw in the previous buses, reduced-seats buses, and taxis protocols by pointing out that key-private public-key encryption schemes should be used instead of ordinary ones.

In a nutshell, this class of protocols initiated by the buses protocol [7] works by routing packets in a way that is independent to the intended receivers. Note that whether this routing strategy is deterministic (*e.g.*, buses) or probabilistic (*e.g.*, DM) does not matter in terms of anonymity.

From this point on, for simplicity, we consider a communication network as a strongly connected (*i.e.*, each node is reachable from any other node) directed graph with N nodes, where packets can only travel along the edges of the graph. For graphs that are not strongly connected, we can always consider the subgraphs containing the nodes connected from each sender node[1].

1.2 Our Results

In view of the existing complicated definitions of anonymity, we make mainly theoretical but also technical contributions.

Theoretically, we present a simple algorithmic syntax which aims to capture a wide class of anonymous communication protocols. We also propose a simple indistinguishability-based definition which captures the strongest possible anonymity known in the literature, namely, unobservability and sender anonymity against malicious receivers [19], simultaneously. The simple syntax and security definition can hopefully make analyzing anonymous communication protocols an easier job. Furthermore, we provide several relaxation of the anonymity notion so that the level of anonymity is still reasonably strong, yet finding efficient constructions is plausible.

Next, we show that the confidentiality of messages and the routing mechanisms can be decoupled, formalizing the idea of Young and Yung [32]. Specifically, we construct anonymous communication protocols generically from a key-private public-key encryption scheme and a new primitive called oblivious forwarding protocol. With this generic approach, we can now focus on constructing the conceptually simpler building block, namely, oblivious forwarding protocols. We then propose a generic construction of oblivious forwarding protocols from any random walk algorithm over graphs.

Our main technical contribution lies in our second construction of oblivious forwarding protocols, which is specially designed for optimizing the probability of successful delivery. This construc-

[1] Also see Appendix A for a discussion on the network environment and deploying our protocols on the internet.

tion ensures that the most "unfortunate" nodes, such as the nodes located in the most isolated areas of the network, receive packets intended for it with at least a fair probability.

We evaluate the optimality by testing our constructions over randomly generated strongly connected graphs, and recording the probability of successful delivery, and number of hops traveled for each sender-receiver pair. Our results show that our optimized protocol performs much better in terms of the probability of successful delivery in realistic networks.

1.3 Technical Overview

We briefly introduce the design of our second construction. Consider a network represented by a strongly connected directed graph, such that packets are routed deterministically according to the routing table stored in each node. Each sender node in the network may not have the complete view of the network. Instead, it only has knowledge of a partial list of intermediate nodes between itself and each receiver node. These partial paths form a tree rooted at the sender node.

Suppose that node i has a packet for node j. Instead of the non-anonymous approach of always sending this packet to the real receiver j directly, it picks a dummy receiver j' according to a distribution independent of the real receiver, and forwards the packet to the dummy according to the routing table. The hope is that the real receiver j is located along the path to the node j'.

The question is then what the distribution of the dummy receivers should be. Intuitively, the nodes that are the least likely to receive the packet are those located at the leaves of the tree. Thus, it is natural to assign the uniform distribution over the set of leaf nodes. Indeed, we show that this distribution is in some sense optimal using standard arguments in linear programming.

1.4 Paper Outline

We first propose in Section 2 (simple yet expressive) syntax and definitions of anonymous communication protocols (AC) and oblivious forwarding protocols (ObF). In Section 3 we then construct AC generically from key-private public-key encryption and ObF. In Section 4, we propose two constructions of ObF, whose performance is evaluated in Section 5. We also briefly discuss practical issues in deploying our AC protocols in Appendix A, and the vast volume of related work in Appendix B. Due to page limitation, preliminary of public-key encryption is deferred to Appendix C.

1.5 Notations

Let λ be the security parameter. All algorithms take 1^λ as input implicitly. Let ϕ be the empty set. Let $[N]$ be the set $\{1, 2, \ldots, N\}$. $P = (p_{kj})_{k,j=1}^{N}$ denotes an N-by-N matrix with the (k, j)-th entry given by p_{kj}. Similarly, $x = (x_i)_{i=1}^{N}$ denotes an N-dimensional (column) vector with the i-th entry given by x_i. If A is a probabilistic algorithm, $x \leftarrow \mathsf{A}(\cdot)$ denotes the computation of x output from A. Let S be a set and $X, Y \sim S$ be distributions over S. Correspondingly, $x \leftarrow S$ denotes the sampling of a uniformly random $x \in S$, and $x \leftarrow X$ denotes the sampling of $x \in S$ according to the distribution X. We denote by $X \approx Y$ that the distributions are identical. Finally, $x := y$ denotes assigning the value of y to the variable x.

2 Formulation of AC and ObF

2.1 Anonymous Communication Protocols

We present a simple yet expressive formulation of anonymous communication protocols. An anonymous communication protocol is run within a network of an arbitrary number of nodes. We consider a dynamic environment where the network topology can change over time, *i.e.*, both nodes and edges may be added or removed. We assume that this network is equipped with a (most likely non-anonymous) routing protocol, so that our anonymous protocol does not need to deal with the changes to the network topology, yet will work regardless of the changes. We model this by letting each participating node in the protocol possess some auxiliary information (*e.g.*, routing tables) maintained by external mechanisms such as the underlying routing protocol.

Informal Description. To participate in the anonymous communication protocol, a node runs the key generation algorithm, without any coordination with any other node, to set up its public and secret keys. It then publishes its public key. We assume that the nodes maintain their auxiliary information (*e.g.*, routing table) and learn the public keys of each other through external mechanisms. For example, they can obtain public keys while learning the network topology using the underlying routing protocol. Alternatively, they might use private information retrieval (PIR) along with a public-key infrastructure to retrieve public keys on-demand yet anonymously (similar to using PIR to retrieve a few IP-addresses of onion-routers in the Tor network on-demand [25]). The participating nodes form a graph G of $N = \mathsf{poly}(\lambda)$ nodes.

Each sender node in the network can encapsulate a message, using its auxiliary information and the public key of the receiver, into a packet ready for forwarding. The creator of the packet or any intermediate node receiving the packet forwards it by running a forwarding algorithm. It takes as input a secret key and some auxiliary information, attempts to decrypt the packet and outputs an outgoing packet and the index of the next hop regardless of whether the decryption is successful. Hopefully, the intended receiver will be one of the intermediate nodes to receive the packet. For anonymity, the packets and the forwarding pattern must not leak any information about the sender and the receiver. It is important to forward the packet regardless of whether the intermediate node happens to be the actual receiver or not. Otherwise, an adversary observing all traffic can notice the disappearance of the packet and discover the real receiver.

Formal Syntax. An anonymous communication protocol $\mathsf{AC} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Fwd})$ is a tuple of PPT algorithms:

- $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$: The probabilistic key generation algorithm is run by a trusted party which setups the network environment. It takes as input the security parameter 1^λ, and outputs a public parameter PP. We note that this is the only algorithm run by a trusted party, and is run once only for setting up the system. Standard practices such as distributed parameter generation can be adopted to reduce trust.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{PP})$: The probabilistic key generation algorithm is run by each node which joins the network individually. It takes as input the public parameter PP, and outputs a public key pk and a secret key sk. The participating nodes form a graph G of $N = \mathsf{poly}(\lambda)$ nodes.
- $p \leftarrow \mathsf{Enc}(j, \mathsf{pk}_j, \mathsf{PK}, m, \mathsf{aux}_k)$: The probabilistic encapsulation algorithm is run by a sender node k. It takes as input a receiver j, its public key pk_j along with some other public keys PK, a message m, and some auxiliary information aux_k of node k, and outputs a packet p.

– $(\{p'_j, j\}_j, M) \leftarrow \mathsf{Fwd}(\mathsf{sk}_k, \mathsf{PK}, P, \mathsf{aux}_k)$: The probabilistic forwarding algorithm is run by a sender node or any intermediate node k. It takes as input a secret key sk of node k, a sequence of public keys PK, a sequence of input packets P, and some auxiliary information aux_k of node k. It outputs a sequence of packets p'_j with a next hop j (or $\perp$), and a sequence of messages M (or $\perp$). Note that the packet p'_j is always forwarded to next hop j (unless $j = \perp$) regardless of whether a valid message $m \in M$ is obtained.

In general, the packet-encapsulation algorithm takes multiple public keys as input, while the packet-forwarding algorithm takes multiple public keys and multiple incoming packets as input. The former captures onion routing protocol and its variants which encrypt messages to a pre-defined route of intermediate routers in layers, while the latter captures for example Mixnets and its variants which shuffle and forward packets in batches. Moreover, the algorithm generates different outgoing packets to multiple next hops. This captures for example some anonymous communication protocols based on broadcasting. Yet, for our purpose, the rest of this paper will stick to the setting where the packet-encapsulation algorithm does not take any extra public keys PK as input, *i.e.*,

$$p \leftarrow \mathsf{Enc}(j, \mathsf{pk}_j, m, \mathsf{aux}_k),$$

while the packet-forwarding algorithm does not take any public keys as input, but only a single incoming packet, and outputs a single outgoing packet and a single next hop, *i.e.*,

$$(p', j, m) \leftarrow \mathsf{Fwd}(\mathsf{sk}_k, p, \mathsf{aux}_k).$$

We note that all the discussions and definitions in the rest of the paper naturally extend to the more general syntax.

Correctness. Roughly speaking, AC is said to be correct, if for any packet generated under an honest execution of the protocol, the packet reaches the intended destination after a reasonable delay with a reasonably high probability. Furthermore, the Fwd algorithm always recovers the message encapsulated in the packet when it reached the intended destination.

It is tricky to formally define correctness. An anonymous communication protocol could have low probability of successful delivery but short expected delivery time when successful, while another could have high probability of successful delivery but long expected delivery time. For the first case, the sender can always re-transmit to make up for the low success probability. Another tricky part is that a protocol might be efficient over some types of graphs but inapplicable to some others. For instance, the buses protocol only works on graphs with a circular path connecting all nodes.

We model this formally by lower-bounding the probability of success delivery after T forwarding by ρ. For any graph G with $N = \mathsf{poly}(\lambda)$ nodes, let $\{k, \mathsf{aux}_k\}_{k=1}^N$ be a set of nodes and auxiliary information. Formally, AC is said to be (T, ρ)-correct on G, if for security parameter $\lambda \in \mathbb{N}$, all sender i, all receiver j, all message m, all public parameter generated by $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$, all key pairs generated by $(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}(\mathsf{PP})$, it holds that

$$\Pr[\mathrm{Correct}_{\mathsf{AC}}(1^\lambda, T, i, j, m, \{k, \mathsf{aux}_k, \mathsf{pk}_k, \mathsf{sk}_k\}_{k=1}^N) = 1] \geq \rho > 0$$

where the probability is taken over the randomness of the experiment $\mathrm{Correct}_{\mathsf{AC}}$ defined in Figure 1a.

Through simple observation, we can conclude that if AC is (T, ρ)-correct on G, then we must have $T \geq l$, where l is the longest of all shortest hop-length between any sender-receiver pair, and that AC must be also (T', ρ')-correct on G for any $T' \geq T$ and $0 < \rho' \leq \rho$. As baselines for comparison, the buses protocol is $(N, 1)$-correct while the broadcast protocol is $(l, 1)$-correct.

$\underline{\text{Exp. } \mathrm{Correct}_{\mathsf{AC}}(1^\lambda, T, i, j, m, \{k, \mathsf{aux}_k, \mathsf{pk}_k, \mathsf{sk}_k\}_{k=1}^N)}$

$t \leftarrow 0, b \leftarrow 0$
$p \leftarrow \mathsf{AC.Enc}(j, \mathsf{pk}_j, m, \mathsf{aux}_i)$
while $t < T$ **then**
  $t \leftarrow t + 1$
  $(p, i, m') \leftarrow \mathsf{AC.Fwd}(\mathsf{sk}_i, p, \mathsf{aux}_i)$
  if $i = j \ \wedge \ m = m'$ **then**
    $b \leftarrow 1$
  endif
endwhile
return b

(a) Experiment for (T, ρ)-correctness of AC

$\underline{\text{Exp. } \mathrm{Correct}_{\mathsf{ObF}}(1^\lambda, T, i, j, \{k, \mathsf{aux}_k\}_{k=1}^N)}$

$t \leftarrow 0, b \leftarrow 0$
$h \leftarrow \mathsf{ObF.Enc}(j, \mathsf{aux}_i)$
while $t < T$ **then**
  $t \leftarrow t + 1$
  $(h, i) \leftarrow \mathsf{ObF.Fwd}(h, \mathsf{aux}_i)$
  if $i = j$ **then**
    $b \leftarrow 1$
  endif
endwhile
return b

(b) Experiment for (T, ρ)-correctness of ObF

Fig. 1: Correctness experiments of anonymous communication (AC) and oblivious forwarding (ObF)

Anonymity. We aim to capture sender and receiver anonymity in the most hostile environment.

For receiver anonymity, we require that a packet leaks nothing about the receiver, neither from the encapsulated message nor the traffic pattern. This implies that a packet encapsulating any message is indistinguishable from each other, so that a sender can safely re-transmit a message or switch to a different message for whatever reasons. Note that the indistinguishability should hold even if the correspondence between messages and senders are known.

For sender anonymity, notice that an adversary observing all traffic must be able to tell the original sender of any packet. Thus, we instead require that when multiple senders send out a set of messages to multiple receivers, no one can tell which message originates from which sender, even if the correspondence between messages and receivers are known.

In technical terms, we consider a security game played between a challenger and a powerful adversary which is able to observe all traffic, corrupt at most all but two of the nodes, and obtain decrypted messages even from non-corrupt nodes. The security game consists of three phases.

In the first phase, the adversary corrupts as many nodes as it wishes, controls and learns from how packets are routed. These are modeled as the corruption and forwarding oracles respectively.

In the second phase, the adversary produces two distinct tuples (*i.e.*, at least one component is different) each consisting of a receiver, a message, and the auxiliary input of a sender. Eventually the challenger is going to create packets according to some parts of the specifications (in the form of sender-message-receiver pairs) of the adversary. So, we also require the adversary to produce a bit to choose that either the sender-message correspondence or the message-receiver correspondence is fixed. This can be thought of as a slot machine with two slots such that the adversary can control the outcome of either one of the slots. If the adversary chooses to fix the sender-message correspondence, an extra restriction is imposed that both of the challenged receivers are not corrupted.

The challenger then picks a random bit to determine the remaining slot of the slot machine: To decide whether the challenge packets should be created according to the specification by the adversary, or the remaining part of the sender-message-receiver pairs should be flipped. For example, if the adversary chose to fix the sender-message correspondence, then the random bit picked by the challenger decides whether the message-receiver correspondence should be flipped, as depicted in Figure 2. The challenger then returns both challenge packets to the adversary.

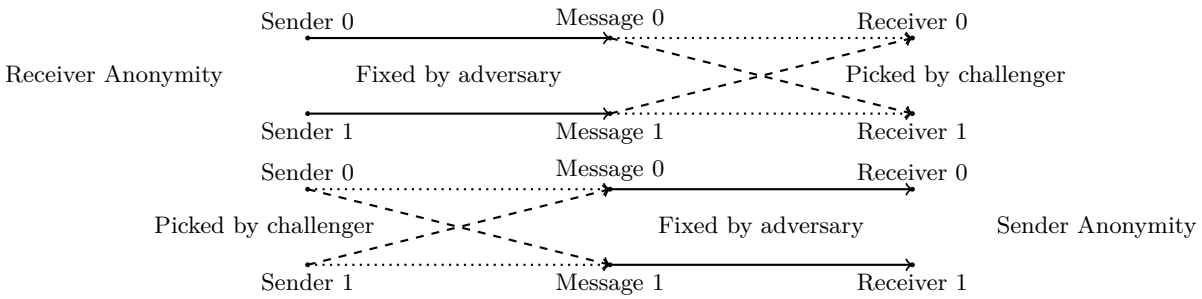


Fig. 2: An illustration of the IND-ANON$^{\mathcal{A}}_{\mathsf{AC}}$ game

In the third phase, the adversary is again given access to the corruption and forwarding oracles. Naturally, if the adversary chose to fix the sender-message pairs, it is still not allowed to corrupt the challenge receivers. Also, when the forwarding oracle is queried with the challenge packets and receivers, no message will be decrypted (but the packet is still forwarded). These are to ensure that the adversary cannot win trivially. Finally, the adversary outputs a bit as a guess of whether the other part of the specification are flipped.

For any graph G with $N = \mathsf{poly}(\lambda)$ nodes, let $\{k, \mathsf{aux}_k\}_{k=1}^N$ be a set of nodes and auxiliary information. Formally, let $\mathsf{Corr}\mathcal{O}$ be a corruption oracle which, on input j, returns sk_j. Let $\mathsf{Fwd}\mathcal{O}$ be a forwarding oracle which, on input (k, p, aux), returns $((p', j), m)$ produced by $\mathsf{AC.Fwd}(\mathsf{sk}_k, p, \mathsf{aux})$. AC is said to have indistinguishability of packets under anonymity attack (IND-ANON), if for any security parameter $\lambda \in \mathbb{N}$, any PPT adversary $\mathcal{A}$ it holds that

$$|2\Pr[\text{IND-ANON}^{\mathcal{A}}_{\mathsf{AC}}(1^\lambda, \{k, \mathsf{aux}_k\}_{k=1}^N) = 1] - 1| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of the experiment and the adversary, and the experiment IND-ANON$^{\mathcal{A}}_{\mathsf{AC}}$ is defined in Figure 3.

Remark. One can fit an onion routing protocol into our definition (by defining an Enc algorithm which encrypts a message to the routers along a random path in layers, and a Fwd algorithm which decrypts the outer-most layer and forwards the inner-layers to the next router). Yet, the routes always terminate at the receiver. An IND-ANON adversary can use the forwarding oracle to figure out the real receiver. So, an onion routing protocol would not satisfy our anonymity requirement.

Relaxations. The above definition takes away almost all quantitative information about the level of anonymity achieved. In practical scenarios, a more efficient protocol with weaker anonymity might be desirable. In this case, we need to know how weak the anonymity guaranteed actually is. To this end, we consider the following reasonable relaxations to our anonymity definition:

- q-bounded Collusion: The above definition essentially bounds the number of corrupted users by $N-2$. In general, we can consider an adversary which is only allowed to corrupt at most q users. The level of anonymity may depend on q. For example, one might consider q as a fraction N.
- 1-out-of-n Anonymity: The above definition models anonymity as a decision problem. Alternatively, we can model it as a search problem to capture anonymity of hiding within a group of $n(\leq N)$ users: In the second phase, the adversary chooses to break either the receiver anonymity

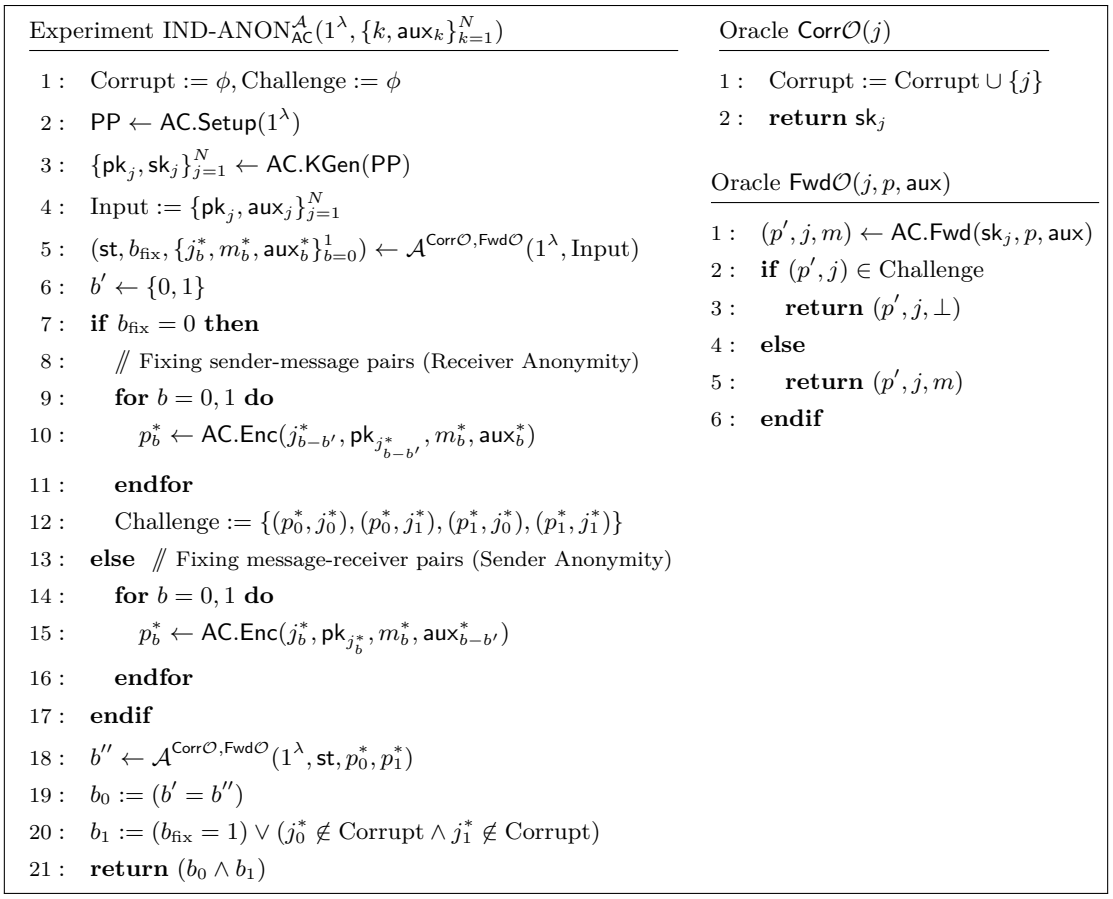


Fig. 3: Experiment for IND-ANON security of anonymous communication protocols

or the sender anonymity. For receiver, it outputs a set of n potential receivers $\{j_i^*\}_{i=1}^n$, a message $m*$, and some auxiliary information of a sender aux^*. For sender, it outputs a target receiver j^*, a message m^*, and a set of auxiliary information for n potential senders $\{\mathsf{aux}_i^*\}_{i=1}^n$. The challenger then picks randomly one of the n senders or receivers and outputs a challenge packet. The adversary wins if it guesses the choice of the challenger correctly.

- CPA-Anonymity: The above definition captures "CCA-anonymity" since the forwarding oracle outputs the trial decryption of the queried packet. We can relax this by letting the forwarding oracle always return $(p', j, \perp)$ (so as to hide the message m encapsulated in the packet).
- Secret Auxiliary Information: The above definition considers an adversary with knowledge of the auxiliary information (*e.g.*, routing table) of all users. In practical setting, one might assume these auxiliary information to be hidden from the adversary.

Relations to Other Notions of Anonymity. We first recall the structure of the definitions by Hevia and Micciancio [20], and those extended by Gelernter and Herzberg [19]. They define a hybrid experiment consisting of a polynomial number of rounds. The experiment is indexed by the type of anonymity attack and a bit b. In each round, the adversary produces two N-by-N matrices $M^{(0)}$ and $M^{(1)}$ where the (i, j)-th entry of the matrices specifies the message sent from node i to node j.

The challenger then executes the anonymous communication protocol on messages contained in $M^{(b)}$. The adversary can choose to continue the experiment, or terminate it by outputting a bit b' as a guess of b. The types of anonymity attacks are captured by imposing different restrictions to the matrices $M^{(0)}$ and $M^{(1)}$. For instance, the unobservability notion is captured by *not* imposing any restriction on the matrices.

Although their definitions is rigorous and expressive, it is considerably more complex than typical security definitions for other cryptographic primitives, such as IND-CPA/CCA security for public-key encryption. The two major complicated aspects are the round-based nature and the unnatural restrictions to the matrices.

Focusing on these two aspects, our definition uses oracles to replace their round-based structure. Our definition also does not restrict the choice of senders, messages and receivers of the adversary. This corresponds to the notion of unobservability. Moreover, we allow the adversary to corrupt the challenged receivers if it chose to fix the message-receiver correspondence. This corresponds to sender anonymity against malicious receivers. We are thus able to capture the two strongest anonymity properties considered in the literature [19].

As our later constructions are inspired by the DM protocol, it is also worth comparing our anonymity definition with that by Young and Yung [32]. Recall that Young and Yung [32] defined (receiver) anonymity and a rather unusual "blocking anonymity", for their Drunk Motorcyclist protocol. While the adversary in the former model is passive, the latter is able to block an arbitrary number of nodes in the network. Both types of adversaries can observe all traffic within the network. However, they are unable to maliciously inject, remove, or modify packets. The goal of the adversary against anonymity is to guess the identity of the real receiver out of all N possible choices, while the goal of the adversary against blocking anonymity is to block any subset of the N nodes so that the real receiver is in this subset.

We make the following observations. From the first glance, blocking anonymity appears to be a generalization of the receiver anonymity. Yet, we observe that their two anonymity notions are somewhat equivalent, up to the size of the blocking set. Suppose there exists an adversary against blocking anonymity, who outputs a set of nodes covering the target receiver with non-negligible probability, we can pick a random member of this set and break anonymity with non-negligible probability as well. Next, we observe that their anonymity definition actually corresponds to our 1-out-of-n anonymity definition. Finally, we remark that Young and Yung [32] did not consider sender anonymity.

2.2 Oblivious Forwarding Protocols

The forwarding pattern of a packet should not depend on the message content but rather the intended receiver. It is natural to separate the routing part of anonymous communication protocols as an independent primitive. We formulate this idea as (receiver-)oblivious forwarding protocols[2].

Informal Overview. An oblivious forwarding protocol is similar to an anonymous communication protocol, except that it only deals with the headers within the packets for routing. Given an intended receiver and some auxiliary information, the Enc algorithm creates a header containing the routing information. The Fwd algorithm creates headers for outgoing packets given an incoming header.

We emphasize again that, as in anonymous communication protocols, regardless of whether the actual receiver is an intermediate node or not, it always forwards the packet to the next hop.

[2] This is not to be confused with the packet-oblivious forwarding protocols in the network community.

Formal Syntax. An oblivious forwarding protocol $\mathsf{ObF} = (\mathsf{Enc}, \mathsf{Fwd})$ is a tuple of probabilistic polynomial-time (PPT) algorithms defined as follows:

- $h \leftarrow \mathsf{Enc}(j, \mathsf{aux}_k)$: The probabilistic encapsulation algorithm is run by a sender node k. It takes as input a receiver j and some auxiliary information aux_k of node k, and outputs a header h.
- $(h', j) \leftarrow \mathsf{Fwd}(h, \mathsf{aux}_k)$: The probabilistic forwarding algorithm is run by a sender node or any intermediate node k. It takes as input an input header h and some auxiliary information aux_k of node k, and outputs a header h and a next hop j.

Correctness. The correctness requirement of oblivious forwarding protocols is essentially the same as that of anonymous communication protocols, except that the former focuses only on the routing aspect.

For any graph G with $N = \mathsf{poly}(\lambda)$ nodes, let $\{k, \mathsf{aux}_k\}_{k=1}^{N}$ be a set of nodes and auxiliary information. Formally, ObF is said to be (T, ρ)-correct on G, if for security parameter $\lambda \in \mathbb{N}$, all sender i, all receiver j, it holds that

$$\Pr[\mathrm{Correct}_{\mathsf{ObF}}(1^\lambda, T, i, j, \{k, \mathsf{aux}_k\}_{k=1}^{N}) = 1] \geq \rho > 0$$

where the probability is taken over the random coins of the experiment $\mathrm{Correct}_{\mathsf{ObF}}$ defined in Figure 1b.

Obliviousness. ObF is said to be oblivious if, for any security parameter $\lambda \in \mathbb{N}$, any pair of receivers j_0 and j_1, and any auxiliary information aux_0 and aux_1, the distributions of the created headers from the Enc algorithm are identical, *i.e.*, $\mathsf{Enc}(j_0, \mathsf{aux}_0) \approx \mathsf{Enc}(j_1, \mathsf{aux}_1)$.

Although we consider perfect obliviousness in this work, one can relax it to statistical or computational obliviousness. We are however unaware of any possible construction, or the potential of efficiency benefits of such constructions. Furthermore, one may also consider alternative definitions, such as a game-based one similar to IND-ANON of anonymous communication protocols.

3 Generic Construction of AC

In this section, we show that anonymous communication protocols can be generically constructed from key-private public-key encryption and oblivious forwarding protocols. Recall that Young and Yung [32] pointed out the need of key-private public-key encryption in several existing anonymous communication protocols, our work here can be seen as formalizing and extending their idea. We also show that the Drunk Motorcyclist protocol is a special case of this generic construction.

Intuitively, our construction works as follows. It encrypts the message to be encapsulated by key-private public-key encryption, and the precedes the ciphertext with the header produced by the oblivious forwarding protocol. To forward a packet, a node attempts to decrypt the ciphertext, and forward the packet using the oblivious forwarding protocol regardless of the decryption result.

3.1 Formal Description

Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme as defined in Appendix C. Let $\mathsf{ObF} = (\mathsf{Enc}, \mathsf{Fwd})$ be an oblivious forwarding protocol as defined in Section 2. Figure 4 presents a generic construction of anonymous communication protocols.

The correctness of this generic construction follows directly from the correctness of the underlying building blocks. The key-privacy of PKE and obliviousness of ObF provides anonymity.

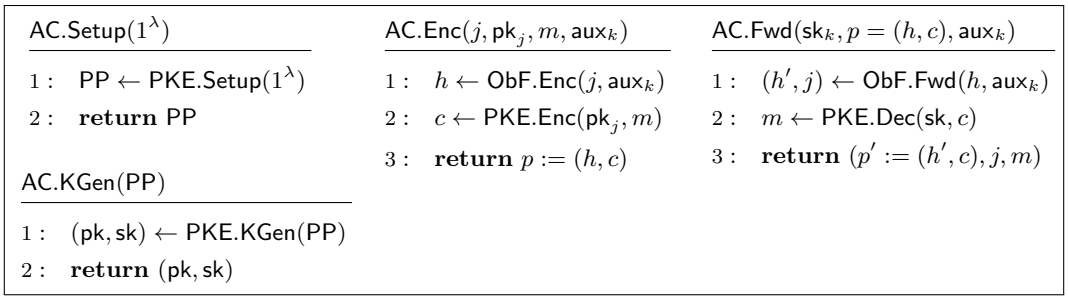


Fig. 4: A generic construction of anonymous communication protocols

Theorem 1. *Assume that* PKE *is correct, and* ObF *is* (T, ρ)*-correct. Then* AC *constructed in Figure 4 is* (T, ρ)*-correct.*

Theorem 2. *Assume that* PKE *is* (IK-CCA2)*-secure, and* ObF *is oblivious. Then* AC *constructed in Figure 4 is* (IND-ANON)*-secure.*

While Theorem 1 is trivial, Theorem 2 is intuitive: Key-private PKE hides the messages and their receivers encapsulated in the packets, so that no adversary can deduce any information compromising anonymity from ciphertexts. Thus, the only way to break anonymity is to observe or control the routing pattern. However, as the headers produced by ObF is independent to its senders and receivers, the headers do not help the adversary in any way either. On the other hand, it is interesting that the seemingly complicated anonymity requirement of AC can be met by the one-line obliviousness requirement of ObF.

Proof. (Theorem 2) Suppose $\mathcal{A}$ is a PPT adversary against the IND-ANON-security of the anonymous communication protocol. We wish to construct a PPT adversary $\mathcal{B}$ against the key-privacy of PKE. For this, we define the hybrids $\mathsf{Hyb}_{b'}$ for $b' = 0$ and 1 as follows.

$\mathsf{Hyb}_{b'}$:

- $\mathcal{B}$ setups the network environment as a graph G with N nodes. Let $\{k, \mathsf{aux}_k\}_{k=1}^N$ be a set of nodes and auxiliary information. It receives PP and pk_j for $j \in [N]$ from the key-privacy challenger. It sends $\{\mathsf{pk}_j, \mathsf{aux}_j\}_{j=1}^N$ to $\mathcal{A}$.
- $\mathcal{B}$ answers queries to the $\mathsf{Corr}\mathcal{O}$ oracle of IND-ANON by redirecting the request to the $\mathsf{Corr}\mathcal{O}$ oracle of IK-CCA2. For queries (j', p', aux') to $\mathsf{Fwd}\mathcal{O}$, $\mathcal{B}$ parses $p' = (h', c')$, redirects (j', c') to the $\mathsf{Dec}\mathcal{O}$ oracle of IK-CCA2 and obtains m. On the other hand, $\mathcal{B}$ runs $(h, j) \leftarrow \mathsf{ObF.Fwd}(h', \mathsf{aux}')$ and returns $((h, c'), j, m)$ to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs $\{(j_b^*, m_b^*, \mathsf{aux}_b^*\}_{b=0}^1$ and b_{fix}. $\mathcal{B}$ picks a bit $\eta \leftarrow \{0, 1\}$. We think of b' (index of the hybrid) and η as two factors to determine whether the specifications of $\mathcal{A}$ should be flipped.
- $\mathcal{B}$ reacts differently for the cases $b_{\mathrm{fix}} = 0$ and $b_{\mathrm{fix}} = 1$:
  - If $b_{\mathrm{fix}} = 0$, meaning that $\mathcal{A}$ chooses to challenge the receiver anonymity, $\mathcal{B}$ sends (j_0, j_1, m_η) to the key-privacy challenger and receives from it a ciphertext c_η^*. Recall that b' is the index of the hybrid. To simulate the header, it runs $h_\eta^* \leftarrow \mathsf{ObF.Enc}(j_{\eta-b'}^*, \mathsf{aux}_\eta^*)$. To simulate the other packet, it runs $c_{1-\eta}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{j_{1-\eta-b'}}, m_{1-\eta})$ and $h_{1-\eta}^* \leftarrow \mathsf{ObF.Enc}(j_{1-\eta-b'}^*, \mathsf{aux}_{1-\eta}^*)$.

- If $b_{\text{fix}} = 1$, meaning that $\mathcal{A}$ chooses to challenge the sender anonymity, $\mathcal{B}$ computes $c_b^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{j_b^*}, m_b^*)$ for $b = 0, 1$. It runs $h_b^* \leftarrow \mathsf{ObF.Enc}(j_b^*, \mathsf{aux}_{b-\eta-b'}^*)$ for $b = 0, 1$.

– Finally, $\mathcal{B}$ sends (p_0^*, p_1^*) where $p_b^* = (h_b^*, c_b^*)$ to the adversary $\mathcal{A}$. $\mathcal{B}$ answers queries to $\mathsf{Corr}\mathcal{O}$ and $\mathsf{Fwd}\mathcal{O}$ as before.

– The game terminates as the adversary $\mathcal{A}$ outputs a guess ξ. The adversary $\mathcal{A}$ wins if $\xi = \eta$.

We differentiate the cases between $b_{\text{fix}} = 0$ and $b_{\text{fix}} = 1$.

Case 1: $b_{\text{fix}} = 0$ There are two differences between Hyb_0 and Hyb_1.

The first difference is that, $\mathcal{B}$ computes $h_{b-\eta}^*$ from $j_{b-\eta}$ in Hyb_0 and from $j_{1-b-\eta}$ in Hyb_1. By the obliviousness of ObF, the two methods of generating $h_{b-\eta}^*$ are indistinguishable in the view of $\mathcal{A}$.

The second difference is that, $\mathcal{B}$ computes $c_{1-\eta}^*$ from $\mathsf{pk}_{j_{1-\eta}}$ in Hyb_0 and from pk_{j_η} in Hyb_1. Since η is chosen at random, the choice of m_η which is directed to the encryption oracle of the key-privacy challenger is random in the view of $\mathcal{A}$. Suppose $\mathcal{A}$ can distinguish between the two methods of generating $c_{1-\eta}^*$ with non-negligible advantage, then it has the same advantage in distinguishing the two methods of generating c_η^*, which breaks the IK-CCA2-security of PKE.

Case 2: $b_{\text{fix}} = 1$ The only difference between Hyb_0 and Hyb_1 is that, $\mathcal{B}$ computes h_b^* from $\mathsf{aux}_{b-\eta}^*$ in Hyb_0 while it computes h_b^* from $\mathsf{aux}_{1-b-\eta}^*$ in Hyb_1. By the obliviousness of ObF, the two methods of generating h_b^* are indistinguishable in the view of $\mathcal{A}$.

Therefore, in either case, Hyb_0 and Hyb_1 are indistinguishable in the view of $\mathcal{A}$. Moreover, in case 2, the bit b' is information theoretically hidden from $\mathcal{A}$ by the obliviousness of ObF. Thus the advantage of $\mathcal{A}$ is zero. Suppose that $\mathcal{A}$ chooses $b_{\text{fix}} = 0$. Denote the random bit chosen by the key-privacy challenger by b_{PKE}. When $b' = b_{\mathsf{PKE}} - \eta$, which occurs with probability $\frac{1}{2}$, $\mathsf{Hyb}_{b'}$ is a perfect simulation of the IND-ANON security game. Conditioned on the above, suppose $\mathcal{A}$ breaks the IND-ANON security of AC with ϵ advantage, then $\mathcal{B}$ also has ϵ advantage in breaking the key-privacy of PKE. □

3.2 Recasting The DM Protocol

Recall that in the DM protocol [32], a node encrypts its message to the intended receiver using a key-private public-key encryption scheme, and forwards the ciphertext to a random neighboring node. Upon receipt of a packet, a node copies the packet to a decryption queue and forwards the packet again to a random neighboring node. For each packet, this process is repeated until its time-to-live (TTL) value vanishes. Straightforwardly, the DM protocol can be seen as an anonymous communication protocol constructed from the above generic approach using an oblivious forwarding protocol $\mathsf{DM\text{-}ObF} = (\mathsf{Enc}, \mathsf{Fwd})$ defined in Figure 5a. The executions of the $\mathsf{DM\text{-}ObF.Fwd}$ represent a simple random walk over a graph, at which a packet travels to each neighboring node with equal probability.

We consider two important parameters for random walk algorithms — the *hitting time*, which is the maximum of the expected time for traveling from any starting node to any destination), and *cover time*, which is the maximum of the expected time for traveling from any starting node to all other nodes at least once. It is well known that the hitting time and cover time of the simple random walk algorithm on general graphs, without using any topological information of the graph, are both $O(N^3)$ [10].

In the context of anonymous communication, there seems to be no reason to avoid using any topological information of the graph. On the contrary, the sender node should exploit this information as much as possible to improve the expected hitting time or probability of successful delivery given a fixed TTL value, as long as its routing strategy remains oblivious.

Intuitively, using a simple random walk algorithm and a fixed TTL value, it is less likely for a node in the more isolated area of a network to receive packets. This motivates us to design new routing strategies that make use of the topological information to improve efficiency.

4 Constructions of ObF

With our generic construction, in this section we aim to construct oblivious forwarding protocols which exploit the topological information to improve efficiency. We first state a generic construction given any transition probability matrix and routing tables of the network. From this generic construction, we can plug in the transition probability matrices of any random walk algorithms over graphs to obtain a class of oblivious forwarding protocols. For demonstration, we use the simple random walk algorithm and the β-random walk algorithm by Ikeda *et al.* [23] as examples.

Next, by introducing a convenient representation of the routing paths from a node as a "connectivity matrix" which is computed using partial topological information, we present our construction which maximizes the minimum probability of successful delivery over all potential receivers for each fixed TTL value in one round, *i.e.*, during the transmission from one dummy receiver to another.

4.1 Generic Construction

Consider a network represented by a strongly connected directed graph. Typically, routing in a such a network is performed in a distributed manner: Each node k maintains its routing table T^k mapping each destination to a next hop. Our strategy works as follows. Regardless of the intended destination, the sender node k chooses a dummy destination $\bar{j}$ according to some distribution independent of the intended destination. The sender node and all intermediate nodes then just route the packet as a normal (non-anonymous) packet to the dummy destination $\bar{j}$. When the packet reaches the dummy destination $\bar{j}$, node $\bar{j}$ chooses another dummy destination as long as the TTL value is still positive.

Formally, let $P = (p_{kj})_{k,j=1}^{N}$ where $p_{kj} \geq 0$, $\sum_j p_{kj} = 1$, $k, j \in [N]$ be any transition probability matrix, and $\mathcal{T} = \{T^k\}_{k=1}^{N}$. Figure 5b defines the oblivious forwarding protocol $(P, \mathcal{T})$-$\mathsf{ObF} = (\mathsf{Enc}, \mathsf{Fwd})$.

The construction in Figure 5b is clearly oblivious as the header h output by Enc and the routing pattern (h, j) output by Fwd is independent to the receiver j. We defer the correctness analysis for specific constructions.

4.2 Construction from any Random Walks

For a node k, $\mathsf{deg}(k)$ and $\mathcal{N}(k)$ are the out-degree and set of neighboring nodes of k respectively. It is obvious that the Drunk Motorcyclist protocol is a special case of the above generic construction, as stated in Lemma 1.

$\underline{\mathsf{ObF.Enc}(j, \mathsf{aux}_k = \mathcal{N}(k))}$

1 : **return** L

$\underline{\mathsf{ObF.Fwd}(h = \ell, \mathsf{aux}_k = \mathcal{N}(k))}$

1 : **if** $\ell = 0$ **then**
2 : **return** $\perp$
3 : **endif**
4 : $j \leftarrow \mathcal{N}(k)$
5 : **return** $(\ell - 1, j)$

(a) DM-ObF: A construction from DM [32], where $\mathcal{N}(k)$ denotes the set of neighboring nodes of node k

$\underline{\mathsf{ObF.Enc}(j, \mathsf{aux}_k = (k, (p_{kj})_{j=1}^N, T^k))}$

1 : **return** $h := (k, L)$

$\underline{\mathsf{ObF.Fwd}(h = (\bar{j}, \ell), \mathsf{aux}_k = (k, (p_{kj})_{j=1}^N, T^k))}$

1 : **if** $\ell = 0$ **then**
2 : **return** $\perp$
3 : **elseif** $\bar{j} = k$ **then**
4 : $\bar{j} \leftarrow (p_{kj})_{j=1}^N$
5 : **endif**
6 : $j := T^k[\bar{j}]$
7 : **return** $((\bar{j}, \ell - 1), j)$

(b) $(P, \mathcal{T})$-ObF: Our generic construction, where $P = (p_{kj})_{k,j=1}^N$ denotes a transition probability matrix, $\mathcal{T} = \{T^k\}_{k=1}^N$ denotes a set of routing tables

Fig. 5: Constructions of oblivious forwarding protocols (L denotes a constant TTL value)

Lemma 1. *Let* $P_{\mathrm{sim}} = (p_{kj})_{k,j=1}^N$ *and* $\mathcal{T}_{\mathrm{sim}} = \{T^k\}_{k=1}^N$ *be defined as*

$$p_{kj} = \begin{cases} \mathsf{deg}^{-1}(k) & \text{if } j \in \mathcal{N}(k) \\ 0 & \text{otherwise} \end{cases}$$

and $T^k[\bar{j}] = \bar{j}$ $\forall \bar{j}$ *respectively. Then* DM-ObF $= (P_{\mathrm{sim}}, \mathcal{T}_{\mathrm{sim}})$-ObF.

Alternatively, the transition probabilities may depend on the local topological information, in particular, the degrees of the neighboring nodes. We consider the β-random walk algorithm designed by Ikeda *et al.* [23].

Definition 1. *The transition probability matrix* $P_\beta = (p_{kj})_{k,j=1}^N$ *of the* β*-random walk algorithm is defined as follows:*

$$p_{kj} = \begin{cases} \frac{\mathsf{deg}^{-\beta}(j)}{\sum_{u \in \mathcal{N}(k)} \mathsf{deg}^{-\beta}(u)} & \text{if } j \in \mathcal{N}(k) \\ 0 & \text{otherwise} \end{cases}$$

The β-random walk algorithm has hitting time and cover time equal to $O(N^2)$ and $O(N^2 \log N)$ respectively on general graphs when $\beta = \frac{1}{2}$ [23]. Therefore, in theory β-ObF $:=$ $(P_\beta, \mathcal{T}_{\mathrm{sim}})$-ObF is asymptotically more efficiency than DM-ObF.

For random walks with transition probability matrix $P = (p_{kj})_{k,j=1}^N$, the (k,j)-th entry of P^ℓ gives the probability of reaching j from k in exactly ℓ steps. Thus, the (k,j)-th entry of $\sum_{\ell=1}^T P^\ell$ gives the probability of reaching j from k in less than T steps. We therefore formulate the correctness of oblivious forwarding protocols induced from random walks as follows.

Theorem 3. *Let* $T > 0$ *be a positive integer. Let* $\rho := \min_{k,j} \sum_{\ell=1}^T P^\ell$ *where* $P = (p_{kj})_{k,j=1}^N$ *is a transition probability matrix. Let* $\mathcal{T} = \{T^k\}_{k=1}^N$ *where* $T^k[\bar{j}] = \bar{j}$ $\forall \bar{j}$*. Then* $(P, \mathcal{T})$-ObF *is* (T, ρ)*-correct.*

4.3 Construction with Optimized Probability of Successful Delivery

Representation of the Routing Paths. To facilitate our discussion, we consider a typical network in which nodes route packets according to routing tables $\mathcal{T}_{\text{opt}}$ built using some distributed shortest path algorithm. Suppose the node k has a partial view of how packets to different destination will be routed. More specifically, it has knowledge of some of the intermediate nodes along the path to each destination. These paths form a tree rooted at node k connecting all other nodes. Using this tree, we construct an N-by-N *connectivity matrix* A^k as follows: If node i is on the path from k to j, set $A^k(i,j) = 1$; Otherwise, set $A^k(i,j) = 0$.

The connectivity matrix A^k features an interesting structure. First, since node j must be on the path from k to j, $A^k(j,j) = 1$ for all j. Second, if node i is a leaf node of the tree, there is only one path from node k to node i. Thus, row i of A^k has only a single '1' which is $A^k(i,i)$. In other words, the 1-norm of row i is 1. Lastly, if node j is a neighbor of node k, there are no intermediate nodes along the path from node k to node j. Thus, column j of A^k has only a single '1' which is $A^k(j,j)$. In other words, the 1-norm of column j is 1.

For conciseness, we will drop the superscript k from A^k and simply write A when the context is clear.

The Construction. We aim to design a routing strategy which is independent to the intended receiver, and maximizes the probability that the most *unfortunate* node receiving its packets in one round. A node is considered the most *unfortunate* if, given a routing strategy, the node receives the packet with the lowest probability.

Formally, consider a sender node with transition probability $x = (x_i)_{i=1}^N$. Let A be the connectivity matrix of the node defined in Section 4.3. Then the i-th entry of Ax indicates the probability that node i belongs to the path from the sender to the dummy destination. Our task is to maximize the minimum of these probabilities, or

$$\begin{aligned} &\max\ (\min_i (Ax)_i) \\ &\text{s.t.}\quad x \geq 0 \wedge \|x\|_1 = 1 \end{aligned}$$

where $x \geq 0$ means $x_i \geq 0\ \forall i \in [N]$.

Intuitively, the optimal solution can be computed as follows. Consider the tree represented by A. Let $\hat{x}$ be the uniform distribution over the set of all leaf nodes. This can be computed by assigning equal weights to the i-th entry of $\hat{x}$ where the i-th row of A contains only a single 1, which is $A(i,i)$ (*i.e.*, node i is a leaf node). This ensures that the most unfortunate nodes (*i.e.*, the leaf nodes) receive their packets with a fair chance. We claim that $\hat{x}$ is an optimal solution to the problem.

Formally, the proposed solution $\hat{x}$ is defined as

$$\hat{x}_i = \begin{cases} \frac{1}{|I|} \text{ if } i \in I \\ 0 \quad \text{otherwise} \end{cases}$$

where $I = \{i : \|A_i\|_1 = 1\}$ and A_i is the i-th row of A.

Remark 1. Interestingly, simple random walk corresponds to assigning equal weights to $\hat{x}_j$ where the j-th *column* (instead of row) of A contains only a single 1, which is $A(j,j)$ (*i.e.*, node j is a neighboring node).

Proof of Optimality. The optimality of $\hat{x}$ is proven via standard arguments in linear optimization. Instead of proving the optimality of $\hat{x}$ directly, which is rather difficult, we construct a dual certificate for the primal solution $\hat{x}$. Then, by the LP Strong Duality Theorem [9, Theorem 4.4], $\hat{x}$ is an optimal solution to (1).

Lemma 2. *The optimal solution to*

$$\begin{aligned} &\max\ (\min_i (Ax)_i) \\ &s.t.\quad x \geq 0 \wedge \|x\|_1 = 1 \end{aligned} \tag{1}$$

where $x \geq 0$ means $x_i \geq 0\ \forall i \in [N]$ is given by

$$\hat{x}_i = \begin{cases} \frac{1}{|I|} \text{ if } i \in I \\ 0 \quad \text{otherwise} \end{cases}$$

where $I = \{i : \|A_i\|_1 = 1\}$ and A_i is the i-th row of A.

Proof. (Lemma 2) The equivalent model of (1) is

$$\begin{aligned} &\min\ -p \\ &\text{s.t.}\ \ Ax \geq pe \wedge e^T x = 1 \wedge x \geq 0 \wedge p \geq 0 \end{aligned} \tag{2}$$

where $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^N$.

To prove such $\hat{x}$ is an optimal solution to (1), we need to find a dual certificate for (2). That is, we need to find an optimal solution for the dual problem.

The dual of the primal problem (2) is

$$\begin{aligned} &\max\ d \\ &\text{s.t.}\quad A^T y \leq -de \wedge e^T y \geq 1 \wedge y \geq 0 \end{aligned} \tag{3}$$

Since $e^T x = 1 \geq 1$ and $x \geq 0$, let $y = \hat{x}$, we have

$$A^T \hat{x} = \sum_{i=1}^{N} A_i^T \hat{x}_i = \sum_{x_i \neq 0} \frac{1}{|I|} A_i^T + \sum_{x_i = 0} 0 \cdot A_i^T = \sum_{x_i \neq 0} \frac{1}{|I|} e_i \leq \frac{1}{|I|} e$$

where A_i is the i-th row of A, e_i is the i-th standard basis vector in $\mathbb{R}^N$.

From the calculation above, we can take $d = -\frac{1}{|I|}$. It is trivial that $Ax \geq \frac{1}{|I|} e$ and the objective value of (2) is $-p = -\frac{1}{|I|}$. Thus, we have found a feasible solution, $y = \hat{x}$, for the dual problem (3) such that the duality gap is zero, *i.e.*, $d = -p = -\frac{1}{|I|}$. Therefore, by the LP Strong Duality Theorem [9, Theorem 4.4], $\hat{x}$ is an optimal solution to (2), so is (1). □

Finally, we define a transition probability matrix $P_{\text{opt}} = (p_{kj})_{k,j=1}^N$ where $p_{kj} = \hat{x}_j^k\ \forall k, j$, and obtain our optimized scheme $\mathsf{Opt\text{-}ObF} = (P_{\text{opt}}, \mathcal{T}_{\text{opt}})\text{-}\mathsf{ObF}$.

It is not easy to formulate the correctness of this construction, at least not in a clean equation form as in the construction induced from random walks. The difficulties arise from that the dummy receiver chosen according to the transition probability matrix is not a neighbor of the sender. The packet may be forwarded multiple times by intermediate nodes until it reaches the dummy receiver. These intermediate nodes are not captured in the transition probability matrix. Moreover, the hop length to each dummy receiver may be different. Thus, a packet might reach several dummy receivers in an instance, while still not reaching the first dummy receiver in another. We would therefore only give a loose bound about the correctness of this construction.

Theorem 4. *Let l be the hop length of the longest shortest path in the graph G. Let I be the set of leaf nodes in the shortest path tree containing the longest shortest path. Then* Opt-ObF *is* $(l, \frac{1}{|I|})$*-correct.*

4.4 Discussions

Optimality. We do not claim that the construction above is optimal over all possible oblivious forwarding protocols. Rather, it optimizes the probability of successful delivery in one round, *i.e.*, during the transmission from one dummy receiver to another. Indeed, the optimal construction is arguably the one where a packet travels each of the nodes at least once with the minimal distance. Unfortunately, finding such a path is an NP-hard problem known as the traveling salesman problem.

On the other hand, optimizing the probability of successful delivery during the entire life span of a packet is undesirable for the following reasons: 1) The system to be optimized becomes too complicated to analyze, as it involves tensor product of L transition probability matrices, where L is the TTL value; 2) The sender might not trust the others to pick dummy receivers honestly. Thus it has the motivation to do the best it can.

Auxiliary Information. As the sender might not have complete knowledge of the paths to all destinations, some leaf nodes in the tree representing its routing paths may actually be intermediate nodes lying on other paths. Thus, assigning positive weight on these nodes would be a waste of effort. However, as the sender learns more about the paths, its strategy will only become better regarding the probability of successful delivery. We remark that the sender should only use side-channel information, which is independent to the real receivers, to update its auxiliary information. Otherwise, an adversary may be able to infer information about the real receivers from the auxiliary information. Note that the side-channel information can be tampered by the adversary without losing anonymity. This is captured in the anonymity game in Figure 3 by allowing the adversary to query with and submit for challenge any auxiliary information of its choice.

5 Experiments

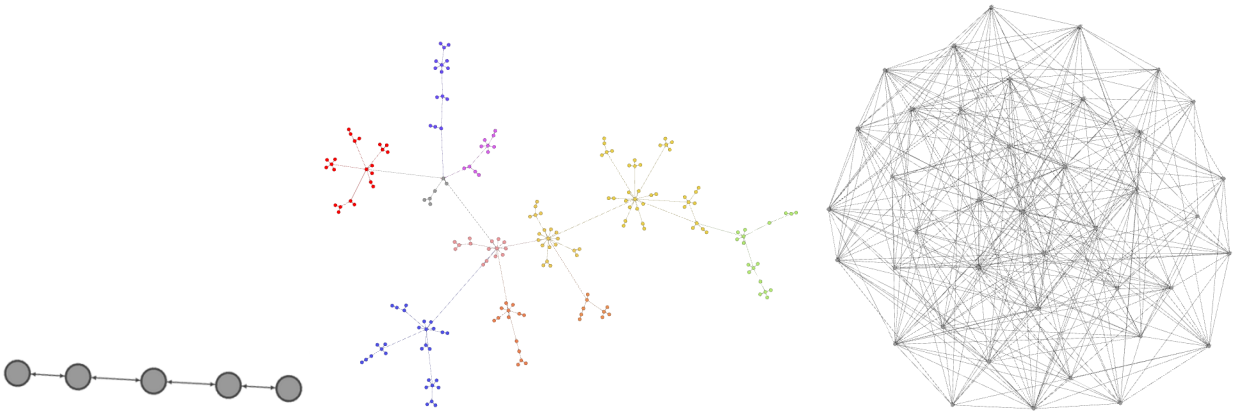
We compare the efficiency of DM-ObF, β-ObF, and Opt-ObF over randomly generated strongly connected directed graphs. For our generic construction, once a dummy receiver is chosen, the packet is routed in the same way as an ordinary packet according to the routing tables. The traffic generated by our protocol is thus the same as the underlying routing protocol defined by the routing tables. We therefore only investigate the average probability of delivering a packet to the intended receiver successfully, and the average number of hops traveled, without performing traffic analysis.

5.1 Graphs Generation

We repeated our experiments for three different types of strongly connected directed graphs, with the number of nodes N set to 2048.

The first type is a simple straight line graph where only node i and $i + 1$ are linked to each other. This serves as an extreme case to show the ineffectiveness of the constructions from some random walk algorithms, in particular, the simple and β-random walks.

The second type is a class of randomly generated graphs according to the *Barabási-Albert (BA) scale-free model* [6]. This model aims to capture the characteristics of real-world networks, which



(a) A straight line graph (b) A BA graph ($n = 200$, $n_0 = 16$, $q = 1$) (c) A dense graph ($n = 40$, $m = 400$)

Fig. 6: Graphs used in our experiments

are often large and sparse. We use the following parameters to generate BA graphs: The initial number of nodes n_0 is set to 16, and the number of new edges q added in each round is 1. Rejection sampling is performed to ensure the chosen BA graphs are strongly connected.

The third type is a class of dense randomly generated graphs, which we call *dense graphs*, using the following method.

> Given the number of nodes N and number of edges M, we first initiate a graph with the N nodes and no edges. For each node k, we maintain a set of connected nodes and a set of unconnected nodes. Initially, only node k is connected to node k itself.
> Then, we perform the following procedures without loss of generality from node 1 to node N: For node k, while there are still unconnected nodes, we pick a random node i in the connected set and a random node j in the unconnected set, and add an edge from i to j. This connects all the nodes (including node k) that are connected to node i to node j, and connects node k to all nodes that node j is connected to. We update the connected and unconnected sets of all the nodes correspondingly.
> Finally, edges are added uniformly at random until the total number of edges reaches M.

We adopt the setting of Young and Yung [32] who set the number of edges M to 32768. This type of graphs is much denser than the graphs representing real world networks, as shown in the toy examples of the three types of graphs in Figures 6a, 6b and 6c respectively.

5.2 Experiment Procedures

We conducted our experiments on 1 straight line graph, 20 randomly generated BA graphs, and 20 randomly generated dense graphs, where each test is repeated for 5 different TTL values $(N/4, N/2, N, 2N, 4N)$. The TTL values are set to contrast with the buses protocol and the broadcast protocol. The former circulates 1 packet and deliver the message with probability 1 in N steps. The latter sends $N - 1$ packets and delivers with probability 1 in l steps, where l is the longest shortest hop length between any sender-receiver pair.

For each graph, 1000 sender-receiver pairs are chosen at random. The sender in each of the pairs runs DM-ObF, β-ObF, and Opt-ObF each for 10 times independently, attempting to deliver

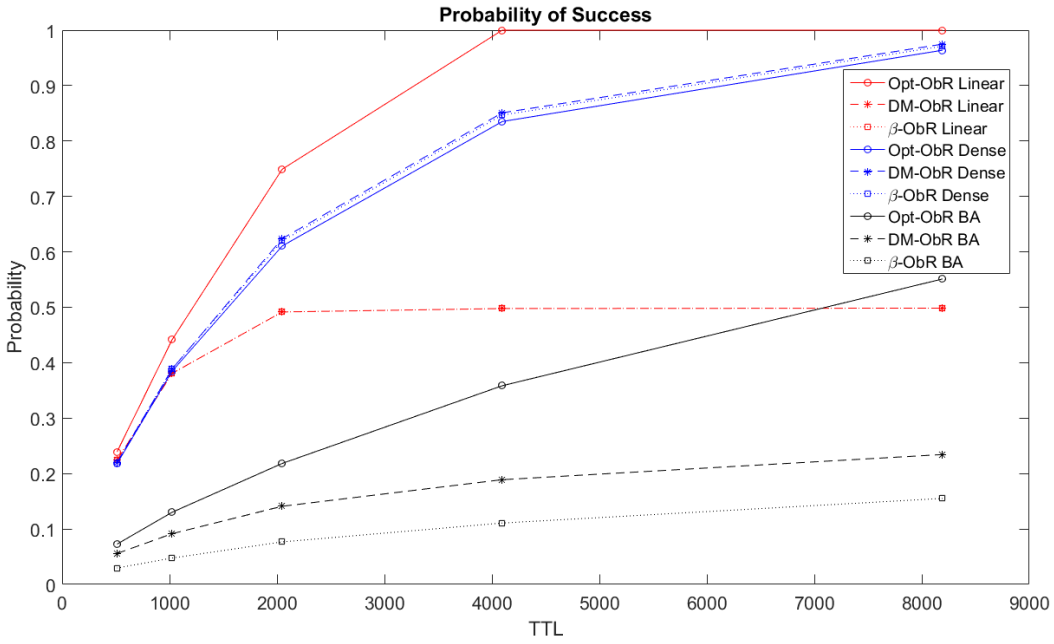


Fig. 7: Mean probability of successful delivery against TTL

a packet to the intended receiver in each instance. The average probability of successful delivery, and number of hops traveled (until the packet is delivered successfully or the TTL reaches 0) are calculated for each combination of graph, protocol, and TTL.

Recall that Opt-ObF requires each node to have partial knowledge about the routing path to each of the other nodes. For simplicity, we assume each node knows the shortest paths to all the other nodes, which are computed using the Floyd-Warshall algorithm.

5.3 Experiment Results

The average probability of successful delivery, and number of hops traveled against the TTL values in each setting are shown in Figure 7, and 8 respectively.

As expected, our Opt-ObF design largely outperforms the other two protocols in terms of the probability of successful delivery in both the straight line graph and BA graphs for all TTL values, which reflects the optimality of our construction. In the dense graph, the differences in probability of successful delivery among the protocols are almost negligible for all TTL values. Our design also has slight advantages in terms of the number of hops traveled by the packets in most of the cases, probably due to the use of shortest paths.

The experiments however did not reflect the theoretical advantage of β-ObF over DM-ObF. A possible explanation is that the hitting time for simple random walks and β-random walks are $O(N^3)$ and $O(N^2)$ respectively, while the TTL values in our experiments are set to $O(N)$. In other words, the packets did not live long enough to tell the tale.

Finally, we remark that in the experiment by Young and Yung [32] for their Drunk Motorcyclist protocol, packets are created by all N nodes in the network constantly until the first packet reaches its intended destination. Therefore, although the average number of hops traveled until successful

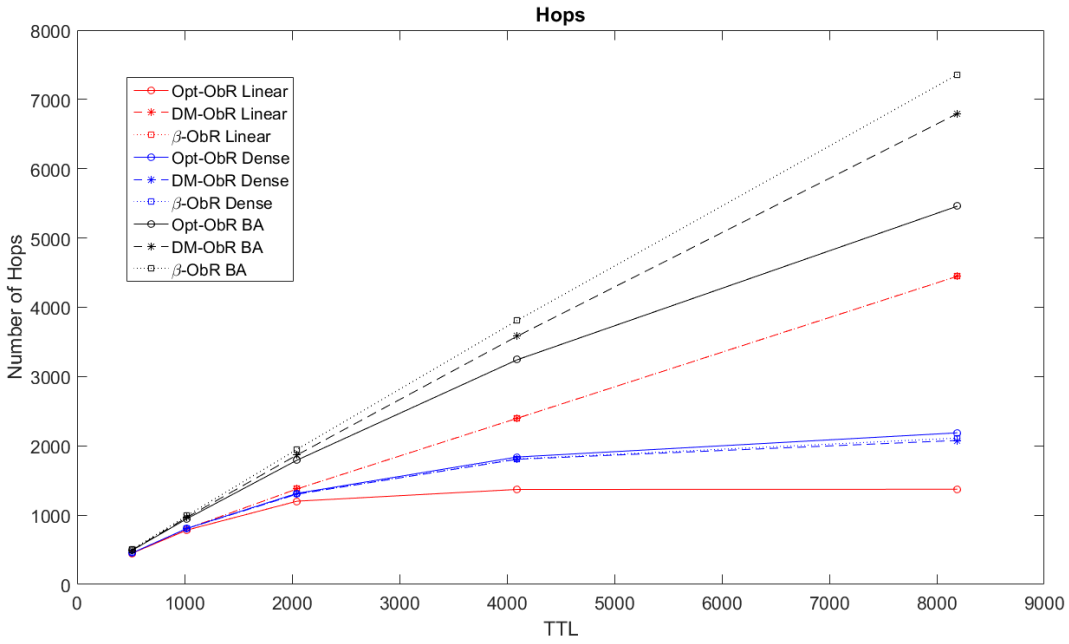


Fig. 8: Average number of hops traveled against TTL

delivery of their protocol in our dense graphs is measured to be 146 according to Young and Yung [32], the number did not reflect the average performance of the protocol as the experiment favored those sender-receiver pairs which are close to each other. Their experiment might have neglected the performance of the protocol for the nodes located in the more isolated areas.

6 Concluding Remark

We have presented simple yet expressive syntax and security definitions of anonymous communication protocols and oblivious forwarding protocols. For the former, we have proposed a generic construction from key-private public-key encryption and oblivious forwarding protocols. For the latter, we have proposed a generic construction from any random walk algorithm over graphs. Our work provides a modular way of constructing anonymous communication protocols and a simple way to analyze their anonymity.

Furthermore, we have specially designed a construction of oblivious forwarding protocols which optimizes the probability of successful delivery. Our experiment results suggest that our optimized construction performs significantly better in terms of the probability of successful delivery in graphs capturing the characteristics of real world networks. In contrast to some efficient yet not-so-anonymous communication protocols, our constructions provide anonymity even in the presence of a powerful adversary, which can observe and control all traffic in the network and corrupt all but two nodes. The strong anonymity comes at an efficiency cost due to the obliviousness of the routing strategy. We leave the study of the trade-off between anonymity and efficiency as a future work.

Acknowledgments

Sherman S.M. Chow is supported by the Early Career Scheme and the Early Career Award of the Research Grants Council, Hong Kong SAR (CUHK 439713).

References

1. Michael Backes, Jeremy Clark, Aniket Kate, Milivoj Simeonovski, and Peter Druschel. Backref: Accountability in anonymous communication networks. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, pages 380–400, 2014.
2. Michael Backes, Ian Goldberg, Aniket Kate, and Esfandiar Mohammadi. Provably secure and practical onion routing. In *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 369–385, 2012.
3. Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. In *2013 IEEE 26th Computer Security Foundations Symposium, New Orleans, LA, USA, June 26-28, 2013*, pages 163–178, 2013.
4. Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. Cryptology ePrint Archive 2014/087, 2014.
5. Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. (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, Scottsdale, AZ, USA, November 3-7, 2014*, pages 513–524, 2014.
6. A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, October 1999.
7. Amos Beimel and Shlomi Dolev. Buses for anonymous message delivery. *J. Cryptology*, 16(1):25–39, 2003.
8. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 566–582, 2001.
9. Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.
10. Graham Brightwell and Peter Winkler. Maximum hitting time for random walks on graphs. *Random Struct. Algorithms*, 1(3):263–276, October 1990.
11. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
12. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.
13. Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 321–338, 2015.
14. Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10*, pages 340–350, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.
15. Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 340–350, 2010.
16. Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. Proactively accountable anonymous messaging in verdict. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 147–162, 2013.
17. Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.
18. Joan Feigenbaum, Aaron Johnson, and Paul F. Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Trans. Inf. Syst. Secur.*, 15(3):14, 2012.
19. Nethanel Gelernter and Amir Herzberg. On the limits of provable anonymity. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 225–236, 2013.

20. Alejandro Hevia and Daniele Micciancio. An indistinguishability-based characterization of anonymous channels. In *Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*, pages 24–43, 2008.
21. Andreas Hirt, Michael J. Jacobson Jr., and Carey L. Williamson. A practical buses protocol for anonymous internet communication. In *Third Annual Conference on Privacy, Security and Trust, October 12-14, 2005, The Fairmont Algonquin, St. Andrews, New Brunswick, Canada, Proceedings*, 2005.
22. Andreas Hirt, Michael J. Jacobson Jr., and Carey L. Williamson. Taxis: Scalable strong anonymous communication. In *16th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2008), Baltimore, Maryland, USA, September 8-10, 2008*, pages 269–278, 2008.
23. Satoshi Ikeda, Izumi Kubo, and Masafumi Yamashita. The hitting and cover times of random walks on finite graphs using local degree information. *Theor. Comput. Sci.*, 410(1):94–100, January 2009.
24. Brian Neil Levine and Clay Shields. Hordes: a multicast-based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.
25. Prateek Mittal, Femi G. Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. PIR-Tor: Scalable anonymous communication using private information retrieval. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*, 2011.
26. Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011*, pages 103–114, 2011.
27. Andreas Pfitzmann and Marit Hansen. 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, August 2010. v0.34.
28. Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998.
29. Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.
30. Ewa Syta, Henry Corrigan-Gibbs, Shu-Chun Weng, David Wolinsky, Bryan Ford, and Aaron Johnson. Security analysis of accountable anonymity in dissent. *ACM Trans. Inf. Syst. Secur.*, 17(1):4:1–4:35, 2014.
31. David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8-10, 2012*, pages 179–182, 2012.
32. Adam L. Young and Moti Yung. The drunk motorcyclist protocol for anonymous communication. In *IEEE Conference on Communications and Network Security, CNS 2014, San Francisco, CA, USA, October 29-31, 2014*, pages 157–165, 2014.

A How to Use Our Anonymous Communication Protocols?

As a quick summary, we have proposed a new formulation of anonymous communication protocols and provided several generic constructions. Note that the functionality of an anonymous communication protocol itself is rather limited: It forwards packets randomly without guaranteeing delivery. We thus provide in the following some guidelines about how to make the best use of these anonymous communication protocols.

Network Environment In the previous sections, we focus our discussion on strongly connected directed graphs, assuming each node in the graph to be both potential sender and receiver, and is willing to forward packets according to the anonymous communication protocol. This setting suffices for networks formed for specific uses (*e.g.*, P2P networks).

In the case of the internet, end users are connected to their local autonomous systems (AS), which are then interconnected to form the internet. While the end users are the potential senders and receivers, they are not supposed to forward packets to neighboring nodes. Instead, the majority of the routing tasks are performed between different ASes. Thus, it is reasonable to think of each AS as a node in an anonymous communication protocol. In this setting, an end user simply sends ordinary

packets to its local AS, which then encapsulates the entire packet as a message using an anonymous communication protocol. Anonymity of the two end users at either side of a communication channel is guaranteed assuming only trusted local ASes of the senders and receivers. We think that this minimal level of trust is acceptable as the end users pay the ASes to subscribe for internet services.

How to Guarantee Successful Delivery? Recall that a packet sent through an anonymous communication protocol is guaranteed to reach its intended destination only with a positive probability. Moreover, since the anonymous communication protocol is sender-anonymous even against malicious receivers, even the intended receiver itself does not know the sender. Thus, there is no indication of success delivery such as the ACK response. The sender can at best send out the message multiple times, using independent randomness for anonymity, so that hopefully the intended receiver can receive the message.

For reliable communication, the sender needs to concatenate its identity to the message, and send out the same message repeatedly, again using independent randomness for anonymity, until receiving an ACK from the receiver. This ACK response is again concatenated by the identity of the receiver, and is sent multiple times so that hopefully the sender can receive it. In short, we can imagine an anonymized TCP connection where each message is sent using an anonymous communication protocol.

Anonymity-Efficiency Trade-off It is of little doubt that anonymous communication protocols with the strongest possible anonymity are inefficient, which is shown under a more complex definition [19]. To benefit from the strong anonymity guarantee, one can consider a hybrid approach in practice: We still use low-latency anonymous communication protocols (*e.g.*, Tor) to setup a fixed route between the sender and the receiver. However, at somewhere in the middle of the route, we use strong anonymous communication protocols within a small subgraph to "cut off" the link. We note that it requires a careful analysis of this approach, perhaps under relaxed definitions, to examine the actual gain of anonymity. We leave it as a future work.

B Related Work

B.1 Other Anonymous Communication Protocols

Anonymous communication protocols can be classified roughly into two categories: One class (which is also our focus) provides strong anonymity. Another class features low latency and scalability, which often rely on trusted participants, servers, or other third parties. Examples in this class include the classical Crowds [29], mix networks (Mixnets) [11], and onion routing [28] (*e.g.*, Tor [17]).

Crowds provides sender anonymity by having the sender randomly forward requests to crowd members, until the request eventually reaches the receiver. Hordes [24] replaces the reply mechanism of Crowds and onion routing by multi-cast to improve efficiency.

Mixnets collect packets from different sources, shuffle and forward them to the next hop in a random order. Multiple layers of encryption is used. The message is in the inner-most layer.

The idea of layered encryption is also applied in onion-routing, where a sender randomly selects a path of "somewhat trusted" routers and encrypts its packet to the routers along this path in layers, so that the last router can send the inner-most content to the intended receiver. The Tor network [17] is the most widely deployed anonymous communication network which uses onion routing as its underlying routing mechanism.

A common problem in most of these schemes (in the case of Mixnets, we consider the efficient variants which do not use zero-knowledge proofs) is that, anonymity is not guaranteed in a strong adversarial model where the adversary is able to observe or even control the traffic. In particular, the sender and receiver will be known to the first and last relay respectively for Tor.

While our work focuses on a highly distributed setting where each user in the network sends and forwards packets individually, some recent work, such as Dissent [14,31] and Riposte [13], utilize cooperation among users to achieve strong anonymity and efficiency at the same time.

Dissent [14,31] introduces semi-trusted servers to make dining cryptographers networks (DC-nets) [12] practical in a decent scale. It provides anonymity if at least one of the servers is honest. Dissent provides also accountability which is not considered in most anonymous communication protocols. However, Dissent ideally assumes that all members remain connected and send correct signed messages during one round. It takes a very long ($O(N)$) time to exclude a single disruptor.

Riposte [13] works on a slightly different setting where a huge number of users wish to post on a shared bulletin board anonymously. Compared to Dissent, Riposte provides similar privacy guarantee, and is able to identify malicious users faster.

B.2 Other Frameworks for Anonymity Analysis

While we focus on giving a simple definition for the strongest possible anonymity, where leakage of anonymity is negligible, Backes *et al.* [3,5,4] formulated a framework AnoA to qualitatively and quantitatively analyze abstract anonymous communication protocols. Their definition is similar to ours in the sense that they also consider a security game played between a powerful adversary and a challenger. Similar to our relaxations, they also introduce adversary classes as wrappers of the powerful adversary to capture realistic attacks. There are several differences from our definitions.

- First, instead of a indistinguishability-style definition which quantifies anonymity leakage additively, they give a differential-privacy-style definition, which quantifies anonymity leakage both multiplicatively and additively. This is helpful for analyzing imperfect anonymous communication protocols which provide weak anonymity. However, we remark that we can easily add the multiplicative factor to our definition as well if desired.
- Second, the anonymous communication protocols in AnoA are modeled abstractly as general interactive Turing machines, whereas in our work we give a simple syntax to capture a wide range of anonymous protocols. We think this makes our anonymity definition easier to use.
- Lastly, AnoA considers static corruption, *i.e.*, the set of corrupted entities is chosen at the beginning of the anonymity game, while we consider adaptive corruption.

Besides the general framework, much effort has been made over the decades to analyze various anonymous communication protocols under different anonymity definitions and adversarial capabilities. Recent examples include a probabilistic analysis to onion routing [18], the analyze of Tor in the UC framework [18,2], fingerprinting attacks on onion routing [26], and accountable anonymous communication [15,31,16,30,1].

C Preliminary

A public-key encryption scheme is a tuple of PPT algorithms $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ defined below.

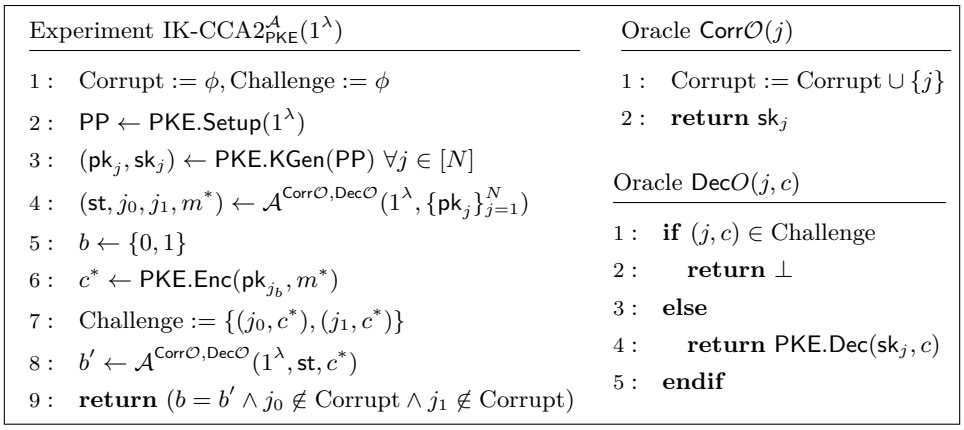


Fig. 9: Experiment for IK-CCA2 security of public-key encryption (modified from Bellare *et al.* [8])

- $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$ is a probabilistic algorithm which takes as input the security parameter λ, and outputs a public parameter PP.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{PP})$ is a probabilistic algorithm which takes as input the public parameter PP, and outputs a public key pk and a secret key sk.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ is a probabilistic algorithm which takes as input the public key pk and a message m, and outputs a ciphertext c.
- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ is a deterministic algorithm which takes as input the secret key sk and a ciphertext c, and outputs a message m.

We say PKE is correct if it holds that

$$\Pr[m' = m : \mathsf{PP} \leftarrow \mathsf{KGen}(1^\lambda); (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{PP}); c \leftarrow \mathsf{Enc}(\mathsf{pk}, m); m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Key-privacy of PKE is introduced by Bellare *et al.* [8], which requires that no probabilistic polynomial time (PPT) adversary can distinguish between ciphertexts produced from two public keys chosen by the challenger. In this work, we consider a slightly modified definition, where the adversary is given access to a corruption oracle which returns the secret key of the requested party.

Let $N = \mathsf{poly}(\lambda)$ be an integer. Let $\mathsf{Corr}\mathcal{O}$ be a corruption oracle which, on input j, returns sk_j. Let $\mathsf{Dec}\mathcal{O}$ be a decryption oracle which, on input j and c, returns $m \leftarrow \mathsf{Dec}(\mathsf{sk}_j, c)$. We say PKE is key-private under adaptive chosen ciphertext attack (IK-CCA2) if, for any PPT adversary $\mathcal{A}$,

$$|2\Pr[\text{IK-CCA2}^{\mathcal{A}}_{\mathsf{PKE}}(1^\lambda) = 1] - 1| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of the experiment and the adversary, and the experiment IK-CCA2$^{\mathcal{A}}_{\mathsf{PKE}}$ is defined in Figure 9. One can also define a corresponding IK-CPA notion for chosen plaintext attack security by removing the decryption oracle.