Another Look at Anonymous Communication

Russell W. F. Lai, Kam-Fung Cheung, Sherman S. M. Chow, *Member, IEEE,*
and Anthony Man-Cho So, *Senior Member, IEEE*

Abstract—Anonymous communication is desirable for personal, financial, and political reasons. Despite the abundance of frameworks and constructions, anonymity definitions are usually either not well defined or too complicated to use. In between are ad-hoc definitions for specific protocols which sometimes only provide weakened anonymity guarantees. 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 to several relaxations. We also propose a modular construction from any key-private public-key encryption scheme, and a new primitive – oblivious forwarding protocols, of which we give two constructions. The first is a generic construction from any random walk over graphs, while the second is optimized for the probability of successful delivery, with experimental validation for our optimization. Anonymity is guaranteed even when the adversary can observe and control all traffic in the network and corrupt most nodes, in contrast to some efficient yet not-so-anonymous protocols. We hope this work suggests an easier way to design and analyze efficient anonymous communication protocols in the future.

Index Terms—anonymous communication, provable anonymity, key-privacy, oblivious forwarding, global adversary

1 Introduction

Since the seminal work of Chaum [1], the notion of anonymous communication (AC) has been extensively studied in the past decades. The goal of AC is to hide the correspondence between senders and receivers of messages, and/or perhaps also their identities. There are plentiful reasons to communicate anonymously, such as to act against censorship and mass surveillance, to protect the privacy of personal preferences, and to express minority opinions. Many cryptographic primitives or systems which provide some sort of sender anonymity (*e.g.*, [2], [3]) often implicitly assume the users to run them on top of an AC network. Using AC has become increasingly popular among the general public, as indicated by the success of the Tor network [4].

1.1 Provable Anonymity against a Global Adversary

Very often, research on AC focuses on achieving low latency, while the anonymity guarantee is not well defined. Pfitzmann and Hansen [5] consolidated informally a collection of terminologies (*e.g.*, unlinkability, anonymity, unobservability) which are commonly used in the literature. Hevia and Micciancio [6] 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 [7] extended the work of Hevia and Micciancio [6] to the setting with adaptive adversaries including malicious receivers. In particular, sender anonymity against malicious receivers is considered the strongest anonymity possible in this setting. Unfortunately, not many of the recent works used these formal definitions: They are too complicated, as admitted by Gelernter and Herzberg [7], 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 AC protocols.

A particular class of AC protocols 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 [8], which a large N-by-N matrix of ciphertexts (the bus) circulates along a fixed route covering all N nodes in the network. The reduced-seats buses [9] and the taxis [10] have improved efficiency upon the buses by reducing the size of the ciphertext carrier. At its extreme, Young and Yung [11] recently proposed the *Drunk Motorcyclist* (DM) 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. They proved that the DM protocol is anonymous in the standard model based on the decisional Diffie-Hellman (DDH) assumption. Young and Yung [11] also pointed out the definitional issues of the buses protocol, cryptanalyzed the reduced-seats buses protocol, and proposed a distinguishing attack against the taxis protocol. These results highlighted the importance of using key-private public-key

- *R. Lai is with the Chair of Applied Cryptography, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany. Part of the work was done while he was with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: russell.lai@cs.fau.de*
- *K.-F. Cheung is with the Institute of Transport and Logistics Studies, The University of Sydney, Australia. Part of the work was done while he was with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong. E-mail: kf.cheung@sydney.edu.au*
- *S. Chow (the corresponding author) is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: sherman@ie.cuhk.edu.hk*
- *A. So is with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong. E-mail: manchoso@se.cuhk.edu.hk*
- *A preliminary version of this work appeared in MyCrypt 2016.*
- *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).*

encryption schemes [12]. In a nutshell, this class of protocols initiated by the buses protocol [8] works by routing packets in a way that is independent of the intended receivers.

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. For other graphs, 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 (AC) protocols. We also propose simple indistinguishability-based definitions which capture the strongest possible anonymity known in the literature, namely, unobservability and sender anonymity against malicious receivers [7]. The simple formulation can hopefully make analyzing AC protocols easier. We further provide several relaxations of the anonymity notion so that the level of anonymity is still reasonably strong, yet finding efficient constructions is plausible.

Formalizing an idea of Young and Yung [11], we show that the confidentiality of messages and the routing mechanisms can be decoupled. Specifically, we construct AC protocols from any key-private public-key encryption scheme and a new primitive called oblivious forwarding (OF) protocol. With this generic approach, now we can focus on constructing OF protocols, a conceptually simpler building block.

We first propose a generic construction of OF protocols from any random walk algorithm over graphs. To send a message, the sender samples a dummy receiver according to the random walk (independent of the real receiver), and forwards the encrypted message to the dummy receiver using an underlying (non-anonymous) communication protocol. Our main technical contribution lies in our specific instantiation of the OF protocols, which is specially designed for optimizing the probability of successful delivery (p_s). We wish to ensure that the most "unfortunate" nodes, *i.e.*, those located in the most isolated areas of the network, receive packets intended for it with at least a fair probability. Intuitively, from a sender perspective, the most "unfortunate" nodes are those located at the leaves of the sender's shortest paths 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.

We evaluate the optimality of our constructions over random strongly connected graphs. We record p_s and the number of hops h traveled for each successfully delivered packet, which are sufficient to calculate the expected time needed for a successful delivery and the expected network capacity consumed[2]. The results show that our optimized protocol performs much better in terms of p_s in realistic networks.

1. Also see the full version for a discussion on the network environment and deploying our protocols on the Internet.

2. See § 8.4 for details.

2 Anonymous Communication (AC)

We present two simple yet expressive formulations of AC protocols – an ad-hoc and a persistent variant[3], under the following minimalistic setup assumptions. An AC protocol is run within a network (a strongly connected directed graph) of an arbitrary number of nodes. The network topology is dynamic and can change over time, *i.e.*, both nodes and edges may be added or removed. The network is equipped with a (most likely non-anonymous) routing protocol, so that our AC 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 k in the protocol possess some auxiliary information aux_k (*e.g.*, routing tables) maintained by some external mechanisms such as the underlying routing protocol.

2.1 Overview

To participate in the AC 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 [13]). The participating nodes form a graph G of N nodes.

How a node encapsulates messages into packets varies in the *ad-hoc* and persistent variants. In the former, each sender node in the network can encapsulate a message directly, using (optionally) its secret key, its auxiliary information and the public key of the receiver, into a packet ready for forwarding. We call this variant ad-hoc since a new packet is produced for each new message to be sent. 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. This variant captures protocols such as onion routing/mixnets [1] and DM [11].

In the *persistent* variant, any node (usually those which just joined the network) can produce an empty packet which is circulated within the network. Typically, once created, these packets will never expire but persistently exist. To encapsulate a message, a sender node must wait for a packet to arrive, then run the forwarding algorithm with the message as an additional input. Similar to the above, the algorithm first attempts to decrypt the packet, yet it merges the new message into the outgoing packet which is then forwarded to the next hop. This captures protocols such as buses [8].

In either case, hopefully, the intended receiver will be one of the intermediate nodes to receive the packet.

3. While it is possible to unify the syntax and anonymity notions to capture both variants, unifying correctness seems to be cumbersome.

2.2 Correctness

Informally, we say that an AC protocol is 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 forwarding algorithm always recovers the message encapsulated in the packet when it reached the intended destination.

This intuitive idea is tricky to formalize. An AC protocol could have a low probability of successful delivery (p_s) but a short expected delivery time when successful, while another could have a high p_s but a long expected delivery time. For the first case, the sender can always retransmit using AC as a black box[4] to make up for the low p_s. 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. Finally, having two variants of AC protocols adds extra complication. Our approach is to model correctness formally by lower-bounding p_s after T rounds of forwarding by ρ.

2.3 Privacy

It is a reasonable expectation that only the intended receiver is able to extract from a packet messages that are encrypted for it. Any third party other than the sender and the receiver should not be able to learn any information about the message from the packet. We formally define privacy similar to the indistinguishability of encryptions for public-key encryption.

2.4 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, even if the correspondence between messages and senders are known. 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. As long as an AC protocol is used as a black box, no external mechanisms (*e.g.*, retransmission, error-catching) can compromise the anonymity of the users.

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 two security games, for sender and receiver anonymity respectively, played between a challenger and a powerful adversary, both consisting of three phases. In the first phase, the adversary is given the security parameter and, in the case of the receiver anonymity game, two honestly generated public keys and access to the forwarding oracle.

In the second phase, the adversary produces the public keys of the other (corrupted) nodes, and two sender-message-receiver tuples which are distinct (*i.e.*, at least one component is different). Eventually, the challenger is going to create packets according to some parts of the specifications of the adversary. Specifically, in the sender anonymity game the message-receiver correspondence is fixed, while in the receiver anonymity game the sender-message 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. 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, in the receiver anonymity game, the random bit picked by the challenger decides whether the message-receiver correspondence should be flipped, as depicted in Fig. 1. The challenger then returns both challenge packets to the adversary.

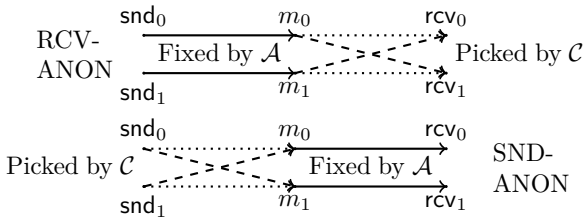

Fig. 1: An illustration of the anonymity games

Lastly, the adversary outputs a bit as a guess of whether the other part of the specification is flipped.

2.5 Ad-Hoc AC (aAC)

We proceed with the formal definitions of ad-hoc AC protocols. The persistent variant will be defined in the full version.

2.5.1 *Syntax.*

An ad-hoc AC protocol $\mathsf{aAC} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Fwd})$ is a tuple of PPT algorithms:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: The probabilistic setup algorithm is run by a trusted party which initiates 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 the trust reliance.

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$: Each node joining the network runs the probabilistic key generation algorithm 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 nodes.

$\{p_k, \mathsf{int}_k\}_k \leftarrow \mathsf{Enc}(\mathsf{sk}_{\mathsf{snd}}, \mathsf{PK}, \{m_j, \mathsf{rcv}_j\}_j, \mathsf{aux}_{\mathsf{snd}})$: The probabilistic encapsulation algorithm is run by a sender node snd to encapsulate messages into packets. It takes as input an (optional) secret key $\mathsf{sk}_{\mathsf{snd}}$ of node snd, the sequence of all public keys PK, a sequence of messages m_j with their intended receiver rcv_j, and some auxiliary information $\mathsf{aux}_{\mathsf{snd}}$ of node snd. It outputs a sequence of outgoing packets p_k with next hops int_k. The packet p_k is forwarded to next hop int_k.

$(\{p_k, \mathsf{int}_k\}_k, M) \leftarrow \mathsf{Fwd}(\mathsf{sk}_{\mathsf{int}}, \mathsf{PK}, P, \mathsf{aux}_{\mathsf{int}})$: The probabilistic forwarding algorithm is run by an intermediate node int to forward packets and to decapsulate packets destined to itself.

4. Re-transmission triggered by events *depending* on the protocol in a non-black-box way might compromise anonymity.

It takes as input a secret key $\mathsf{sk}_{\mathsf{int}}$ of node int, the sequence of all public keys PK, a sequence of incoming packets P, and some auxiliary information $\mathsf{aux}_{\mathsf{int}}$ of node int. It outputs a sequence of outgoing packets p_k with next hops int_k, and a sequence of decapsulated incoming messages M if any (ϕ otherwise). If $p_k \neq \perp$, it is forwarded to next hop int_k regardless of whether any valid messages M are obtained. If the node declines to forward the packet, it outputs $(\perp, M)$.

Remark 1. One can fit an onion routing protocol into our definition. Specifically, we can define 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 adversary against the anonymity can use the forwarding oracle to figure out the real receiver. So, an onion routing protocol would not satisfy our anonymity requirement.

Remark 2. The forwarding algorithm takes multiple public keys (instead of just that of the receiver) and multiple incoming packets as input. The former captures protocols which employ onion routing which encrypts 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 AC protocols based on broadcasting.

2.5.2 Correctness.

For simplicity, we only define correctness for protocols in which the encapsulation algorithm takes as input a single message-receiver pair (m, j) and outputs a single packet p with next hop k. Similarly, we restrict the forwarding algorithm to only take as input one packet and outputs one packet. It is straightforward (but cumbersome) to extend the definition for general protocols.

For any graph G with any N nodes, let $\{\mathsf{aux}_k\}_{k=1}^N$ be their auxiliary information. aAC is said to be (T, ρ)-correct on G if, for all security parameters $\lambda \in \mathbb{N}$, all senders i, all receivers j, all messages m, all public parameters $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, all key pairs $(\mathsf{pk}_k, \mathsf{sk}_k) \in \mathsf{KGen}(\mathsf{pp})$,

$$p_s := \Pr[\mathsf{Corr}^T_{\mathsf{aAC}}(1^\lambda, \mathsf{snd}, \mathsf{rcv}, m, \{\mathsf{aux}_k, \mathsf{pk}_k, \mathsf{sk}_k\}_{k=1}^N) = 1] \geq \rho$$

where the probability is taken over the randomness of the experiment $\mathsf{Corr}^T_{\mathsf{aAC}}$, which is defined in Fig. 2.

We can observe that if aAC 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 aAC 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.

2.5.3 Privacy and Anonymity.

aAC is said to be {private, sender-anonymous, receiver-anonymous} if, for any security parameter $\lambda \in \mathbb{N}$, any PPT adversary $\mathcal{A}$ it holds that

$$|\Pr[\mathsf{Exp}^0_{\mathsf{aAC},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^1_{\mathsf{aAC},\mathcal{A}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

with probability taken over the random coins of $\mathsf{Exp}^b_{\mathsf{aAC},\mathcal{A}}$ for $\mathsf{Exp} \in \{\mathsf{Priv}, \mathsf{Snd\text{-}Anon}, \mathsf{Rcv\text{-}Anon}\}$ which is defined in Fig. 3, 4, and 5, respectively.

3 Oblivious Forwarding (OF)

How a packet is routed should not depend on the message but rather the intended receiver. It is natural to separate the routing part of AC as an independent primitive. We formulate this idea as (receiver-)oblivious forwarding (OF) protocols[5].

3.1 Overview.

An OF protocol is similar to an AC protocol, except that it only deals with the headers of the packets for routing. It consists of the header encapsulation algorithm Enc which creates a header h containing the routing information, and the forwarding algorithm Fwd which creates headers for outgoing packets given an incoming header. Naturally, as AC, we define an ad-hoc variant (here) and a persistent variant (in the full version), which differ in whether Enc or Fwd requires the receiver information as input.

3.2 Ad-Hoc OF (aOF)

3.2.1 Syntax.

An ad-hoc OF protocol $\mathsf{aOF} = (\mathsf{Enc}, \mathsf{Fwd})$ is a tuple of PPT algorithms defined as follows:

$\{h_k, \mathsf{int}_k\}_k \leftarrow \mathsf{Enc}(\{\mathsf{rcv}_j\}_j, \mathsf{aux}_{\mathsf{snd}})$: The probabilistic encapsulation algorithm is run by a sender node snd. It inputs a set of receivers $\{\mathsf{rcv}_j\}_j$ and some auxiliary information $\mathsf{aux}_{\mathsf{snd}}$ of node snd. It outputs a set of headers h_k with a next hop int_k.

$\{h_k, \mathsf{int}_k\}_k \leftarrow \mathsf{Fwd}(H, \mathsf{aux}_{\mathsf{int}})$: The probabilistic forwarding algorithm is run by a sender node or any intermediate node int. It takes as input a set of incoming headers H and some auxiliary information $\mathsf{aux}_{\mathsf{int}}$ of node int. It outputs a set of headers h_k with a next hop int_k. If the node declines to forward the header, it outputs $\perp$.

3.2.2 Correctness.

The correctness requirement of OF protocols is essentially the same as that of AC protocols, except that the former focuses only on the routing aspect.

For any graph G with any N nodes, let $\{\mathsf{aux}_k\}_{k=1}^N$ be their auxiliary information. aOF is said to be (T, ρ)-correct on G if, for all security parameters $\lambda \in \mathbb{N}$, all senders snd, all receivers rcv, it holds that

$$\Pr[\mathsf{Corr}^T_{\mathsf{aOF}}(1^\lambda, \mathsf{snd}, \mathsf{rcv}, \{\mathsf{aux}_k\}_{k=1}^N) = 1] \geq \rho$$

where the probability is taken over the random coins of the experiment $\mathsf{Corr}^T_{\mathsf{aOF}}$, which is defined in Fig. 2.

3.2.3 Obliviousness.

OF is said to be oblivious if, for any security parameter $\lambda \in \mathbb{N}$, any sequences of receivers $\{\mathsf{rcv}_{0,j}\}_j$ and $\{\mathsf{rcv}_{1,j}\}_j$, and any auxiliary information aux, such that $|\{\mathsf{rcv}_{0,j}\}_j| = |\{\mathsf{rcv}_{1,j}\}_j|$, the distributions of the created headers from the Enc algorithm are identical, *i.e.*, $\mathsf{Enc}(\{\mathsf{rcv}_{0,j}\}_j, \mathsf{aux}) \approx \mathsf{Enc}(\{\mathsf{rcv}_{1,j}\}_j, \mathsf{aux})$.

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. Alternative definitions, such as a game-based one similar to the anonymity definitions of AC protocols, are also possible.

5. Not to be confused with packet-oblivious forwarding in the network community.

$\underline{\mathsf{Corr}^T_{\mathsf{aAC}}(1^\lambda, \mathsf{snd}, \mathsf{rcv}, m, \{\mathsf{aux}_k, \mathsf{pk}_k, \mathsf{sk}_k\}_{k=1}^N)}$
$t \leftarrow 0, b \leftarrow 0$
$(p, \mathsf{int}) \leftarrow \mathsf{aAC.Enc}(\mathsf{sk}_{\mathsf{snd}}, \mathsf{PK}, m, \mathsf{rcv}, \mathsf{aux}_{\mathsf{snd}})$
while $t < T$ **then**
  $t \leftarrow t + 1$
  $(p, \mathsf{int}, m') \leftarrow \mathsf{aAC.Fwd}(\mathsf{sk}_{\mathsf{int}}, \mathsf{PK}, p, \mathsf{aux}_{\mathsf{int}})$
  if $\mathsf{int} = \mathsf{rcv} \ \wedge \ m = m'$ **then** $b \leftarrow 1$
endwhile
return b

$\underline{\mathsf{Corr}^T_{\mathsf{aOF}}(1^\lambda, \mathsf{snd}, \mathsf{rcv}, \{\mathsf{aux}_k\}_{k=1}^N)}$
$t \leftarrow 0, b \leftarrow 0$
$(h, \mathsf{int}) \leftarrow \mathsf{aOF.Enc}(\mathsf{rcv}, \mathsf{aux}_{\mathsf{snd}})$
while $t < T$ **then**
  $t \leftarrow t + 1$
  $(h, \mathsf{int}) \leftarrow \mathsf{aOF.Fwd}(h, \mathsf{aux}_{\mathsf{int}})$
  if $\mathsf{int} = \mathsf{rcv}$ **then** $b \leftarrow 1$
endwhile
return b

Fig. 2: Correctness experiments of aAC and aOF

$\underline{\mathsf{Priv}^b_{\mathsf{aAC},\mathcal{A}}(1^\lambda)}$
$\mathsf{Ch} \leftarrow 0, \ \mathsf{pp} \leftarrow \mathsf{aAC.Setup}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{rcv}}, \mathsf{sk}_{\mathsf{rcv}}) \leftarrow \mathsf{aAC.KGen}(\mathsf{pp})$
$(\mathsf{st}, \mathsf{sk}_{\mathsf{snd}}, \hat{\mathsf{PK}}, m_0, m_1, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{Enc}\mathcal{O}, \mathsf{Fwd}\mathcal{O}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{rcv}})$
$\mathsf{Ch} \leftarrow 1, \ \mathsf{PK} := \hat{\mathsf{PK}} \cup \{\mathsf{pk}_{\mathsf{rcv}}\}$
$(p, \mathsf{int}) \leftarrow \mathsf{aAC.Enc}(\mathsf{sk}_{\mathsf{snd}}, \mathsf{PK}, m_b, \mathsf{rcv}, \mathsf{aux})$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}\mathcal{O}, \mathsf{Fwd}\mathcal{O}}(\mathsf{st}, (p, \mathsf{int}))$
return b'

$\underline{\mathsf{Enc}\mathcal{O}(\mathsf{PK}, \{m_j, \mathsf{rcv}_j\}_j, \mathsf{aux})}$ // secret-key schemes only
$\{p_k, \mathsf{int}_k\}_k \leftarrow \mathsf{aAC.Enc}(\mathsf{sk}_{\mathsf{rcv}}, \mathsf{PK}, \{m_j, \mathsf{rcv}_j\}_j, \mathsf{aux})$
return $(\{p_k, \mathsf{int}_k\}_k)$

$\underline{\mathsf{Fwd}\mathcal{O}(\mathsf{PK}, P, \mathsf{aux})}$
$(\{p_k, \mathsf{int}_k\}_k, M) \leftarrow \mathsf{aAC.Fwd}(\mathsf{sk}_{\mathsf{rcv}}, \mathsf{PK}, P, \mathsf{aux})$
if $\mathsf{Ch} = 1$ **then** $M \leftarrow \perp$
return $(\{p_k, \mathsf{int}_k\}_k, M)$

Fig. 3: Experiments for privacy of aAC protocols

$\underline{\mathsf{Snd\text{-}Anon}^b_{\mathsf{aAC},\mathcal{A}}(1^\lambda)}$
$(\mathsf{st}, \mathsf{PK}, \{\mathsf{sk}_i, \{m_{i,j}, \mathsf{rcv}_{i,j}\}_j, \mathsf{aux}_i\}_{i=0}^1) \leftarrow \mathcal{A}(1^\lambda)$
if $|\{m_{0,j}, \mathsf{rcv}_{0,j}\}_j| \neq |\{m_{1,j}, \mathsf{rcv}_{1,j}\}_j|$ **then return** 0
$\{p_{i,k}, \mathsf{int}_{i,k}\}_k \leftarrow \mathsf{aAC.Enc}(\mathsf{sk}_{i\oplus b}, \mathsf{PK}, \{m_{i,j}, \mathsf{rcv}_{i,j}\}_j, \mathsf{aux}_{i\oplus b}), \ i \in \{0,1\}$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \{\{p_{i,k}, \mathsf{int}_{i,k}\}_k\}_{i=0}^1)$
return b'

Fig. 4: Experiments for Snd-Anon security of aAC protocols

4 Remarks on Mixnets and Its Variants

For efficiency and reliability, Mixnets [1] and its variants are great choices of AC, as they feature low latency and guarantee successful delivery. Their anonymity is unfortunately limited.

Consider the following simple Mixnet-like protocol: When a sender node wishes to send a message, it picks ℓ Mixnet relays at random, encrypts its message to the receiver, then further encrypts the ciphertext in layers with the public keys of the relays. It then forwards the ciphertext to the first relay. Each intermediate relay waits until it collects a large enough set of ciphertexts, rerandomizes them, and forwards them to the next relays in random order. Eventually, the last relay forwards the innermost ciphertext to the intended receiver.

This type of protocol is neither sender nor receiver anonymous, even in a weak collusion model where the adversary can corrupt only one node of the network, due to the following injection attack: Recall that in both the sender and receiver anonymity games, the adversary would specify two sender-message-receiver pairs. The challenger flips a coin to decide whether it would interchange the senders in the sender anonymity game, or the receivers in the receiver anonymity game, then outputs two packets partially according to the specification given by the adversary. Each of these packets is forwarded to the respective first relay picked by the challenger, which the adversary can observe. The later thus inject enough dummy packets to the first relay to force it to forward packets to the next relay. Since all packets forwarded by this first relay except for the challenge packet are dummy packets created by the adversary, it learns the second relay intended for the challenge packet. Repeating the above, the adversary can link the senders to the receivers of the challenge packets, hence breaking both sender and receiver anonymity.

An idea of augmenting a Mixnet-like protocol so that it satisfies our strong anonymity definitions is as follows: Consider a bounded corruption model where the adversary is only allowed to corrupt q nodes. In the augmented protocol, each intermediate relay collects $(q+2)$ incoming packets, with a catch: They come from different senders. This can be done without compromising sender anonymity by using a variant of linkable ring signatures [2], [14]: The sender signs the t-th layer ciphertext (counting from outer to inner), the value t which is also used to denote the time-step, and the index of the intermediate relay at which the packet arrives at time t. We require that two signatures are linkable if they are issued by

$\underline{\mathsf{Rcv\text{-}Anon}^{b}_{\mathsf{aAC},\mathcal{A}}(1^\lambda)}$

$\mathsf{Ch} \leftarrow 0,\ \mathsf{pp} \leftarrow \mathsf{aAC.Setup}(1^\lambda)$
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{aAC.KGen}(\mathsf{pp}),\ i \in \{0,1\}$
$(\mathsf{st}, \hat{\mathsf{PK}}, \{\{m_{i,j}, \mathsf{rcv}_{i,j}\}_j, \mathsf{aux}_i\}_{i=0}^{1}) \leftarrow \mathcal{A}^{\mathsf{Enc}\mathcal{O},\mathsf{Fwd}\mathcal{O}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{pk}_1)$
if $|\{m_{0,j}, \mathsf{rcv}_{0,j}\}_j| \neq |\{m_{1,j}, \mathsf{rcv}_{1,j}\}_j|$ **then return** 0
$\mathsf{Ch} \leftarrow 1,\ \mathsf{PK} := \{\mathsf{pk}_0, \mathsf{pk}_1, \hat{\mathsf{PK}}\}$
$\{p_{i,k}, \mathsf{int}_{i,k}\}_k \leftarrow \mathsf{aAC.Enc}(\mathsf{sk}_i, \mathsf{PK}, \{m_{i,j}, \mathsf{rcv}_{i\oplus b,j}\}_j, \mathsf{aux}_i),\ i \in \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}\mathcal{O},\mathsf{Fwd}\mathcal{O}}(\mathsf{st}, \{\{p_{i,k}, \mathsf{int}_{i,k}\}_k\}_{i=0}^{1})$
return b'

$\underline{\mathsf{Enc}\mathcal{O}(i, \mathsf{PK}, \{m_j, \mathsf{rcv}_j\}_j, \mathsf{aux})}$

// secret-key schemes only
$\{p_k, \mathsf{int}_k\}_k \leftarrow \mathsf{aAC.Enc}(\mathsf{sk}_i, \mathsf{PK}, \{m_j, \mathsf{rcv}_j\}_j, \mathsf{aux})$
return $(\{p_k, \mathsf{int}_k\}_k)$

$\underline{\mathsf{Fwd}\mathcal{O}(i, \mathsf{PK}, P, \mathsf{aux})}$

$(\{p_k, \mathsf{int}_k\}_k, M) \leftarrow \mathsf{aAC.Fwd}(\mathsf{sk}_i, \mathsf{PK}, P, \mathsf{aux})$
if $\mathsf{Ch} = 1$ **then** $M \leftarrow \perp$
return $(\{p_k, \mathsf{int}_k\}_k, M)$

Fig. 5: Experiment for Rcv-Anon security of aAC protocols

the same signer and are certifying the same time and index of the intermediate relay. Intuitively, since the adversary is only able to inject at most q packets from unique senders, at least 2 packets out of the $(q+2)$ are honestly generated and look random in the view of the adversary. However, for the formal security proof to go through, the two challenge packets have to be simultaneously forwarded to the same relay at one of the time-steps, which is unlikely. We thus leave constructing a strongly anonymous Mixnet-like protocol as an open problem.

5 Generic Constructions of Ad-Hoc AC

We show that ad-hoc AC protocols can be generically constructed from key-private public-key encryption (PKE) and ad-hoc OF protocols. For the construction of the persistent variant, see the full version. Recall that Young and Yung [11] pointed out the need of key-private PKE in several existing AC protocols, our work here can be seen as formalizing and extending their idea. We also show that the DM protocol is a special case of our generic ad-hoc construction. Since the obliviousness of the OF protocols is information-theoretic, by plugging in an existing key-private PKE secure under some intractability assumption into the generic construction, we obtain an AC protocol secure under the same assumption.

We first focus on protocols which send messages one at a time, *i.e.*, the algorithms take as input a message and a receiver (m, rcv) instead of a sequence of them. The constructions can be extended to support multiple messages easily, yet with little performance gain. In § 7, we will discuss a non-blackbox construction of AC protocols which exploit multicasting to gain efficiency.

5.1 Overview

Intuitively, our ad-hoc construction works as follows. It encrypts the message by key-private PKE, and then precedes the ciphertext with the header produced by the OF protocol. To forward a packet, a node attempts to decrypt the ciphertext, and forward the packet using the OF protocol regardless of the decryption result.

While there are plenty of PKE with fairly efficient decryption (*e.g.*, consisting of several exponentiations), it may still be considered expensive for every node to attempt decryption for every received packet. To alleviate the cost, one may choose specific instantiations of the encryption scheme, *e.g.*, Cramer-Shoup [15] or its variants, in which the decryption algorithm first tests the "validity tag" of the ciphertext, and only performs the final decryption if the test is passed.

5.2 Formal Description

Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme. Let $\mathsf{aOF} = (\mathsf{Enc}, \mathsf{Fwd})$ be an ad-hoc OF protocol as defined in § 3. Fig. 6 presents a generic construction of ad-hoc AC protocols.

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

Theorem 5.1. Assume that PKE is correct and aOF is (T, ρ)-correct (§ 3.2.2). Then aAC constructed in Fig. 6 is (T, ρ)-correct (§ 2.5.2).

Theorem 5.2. If PKE is IND-CCA1-secure then aAC in Fig. 6 is private (§ 2.5.3).

We omit the proofs of the above trivial theorems.

Theorem 5.3. Assume that aOF is oblivious (§ 3.2.3). Then aAC in Fig. 6 is sender-anonymous (§ 2.5.3).

Intuitively, the only information about the sender from the output of the encapsulation algorithm is the header and the next hop, which are produced by the OF protocol. Sender anonymity then follows from the fact that the sender-receiver correspondence is hidden by obliviousness.

Formally, we argue that for any PPT adversary $\mathcal{A}$, we have $|\Pr[\mathsf{Snd\text{-}Anon}^{0}_{\mathsf{aAC},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Snd\text{-}Anon}^{1}_{\mathsf{aAC},\mathcal{A}}(1^\lambda) = 1]| = 0$. In the experiment $\mathsf{Snd\text{-}Anon}^{b}_{\mathsf{aAC},\mathcal{A}}$, $\mathcal{A}$ outputs a set of public keys PK, two secret keys sk_i, two message-receiver pairs (m_i, rcv_i), and two auxiliary information aux_i, for $i = 0, 1$. The challenger computes $(h_i, \mathsf{int}_i) \leftarrow \mathsf{aOF.Enc}(\mathsf{rcv}_i, \mathsf{aux}_{\mathsf{snd}_{i\oplus b}})$. It then computes $c_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{rcv}_i}, m_i)$ and returns $p_i := (h_i, c_i)$ to $\mathcal{A}$. Let $\chi_{i,j}$ be the distribution of $\mathsf{aOF.Enc}(\mathsf{rcv}_j, \mathsf{aux}_i)$. By the obliviousness of aOF, $\chi_{0,0}$ has the same distribution as $\chi_{1,0}$, while $\chi_{0,1}$ has the same distribution as $\chi_{1,1}$. Combining both, we have that $(\chi_{0,0}, \chi_{1,1})$ (for case $b = 0$) has the same distribution as $(\chi_{0,1}, \chi_{1,0})$ (for case $b = 1$). The bit b is thus information-theoretically hidden from the view of $\mathcal{A}$. Therefore, we conclude that $\mathcal{A}$ cannot perform better than randomly guessing.

Theorem 5.4. If PKE is IK-CCA1-secure and aOF is oblivious (§ 3.2.3), then aAC constructed in Fig. 6 is receiver-anonymous (§ 2.5.3).

Intuitively, while sender anonymity follows from obliviousness, receiver anonymity additionally requires key privacy

$\underline{\mathsf{aAC.Setup}(1^\lambda)}$
return $\mathsf{pp} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$

$\underline{\mathsf{aAC.KGen}(\mathsf{pp})}$
return $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KGen}(\mathsf{pp})$

$\underline{\mathsf{aAC.Enc}(\mathsf{sk}_{\mathsf{snd}}, \mathsf{PK}, m, \mathsf{rcv}, \mathsf{aux}_{\mathrm{snd}})}$
parse PK **as** $(\ldots, \mathsf{pk}_{\mathsf{rcv}}, \ldots)$
$(h, \mathsf{int}) \leftarrow \mathsf{aOF.Enc}(\mathsf{rcv}, \mathsf{aux}_{\mathsf{snd}})$
$c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{rcv}}, m)$
$p := (h, c)$
return (p, int)

$\underline{\mathsf{aAC.Fwd}(\mathsf{sk}_{\mathsf{int}}, \mathsf{PK}, p, \mathsf{aux}_{\mathsf{int}})}$
parse p **as** (h, c)
$(h', \mathsf{int}') \leftarrow \mathsf{aOF.Fwd}(h, \mathsf{aux}_{\mathsf{int}})$
$m \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{int}}, c)$
if $(h', \mathsf{int}') = (\bot, \bot)$ **then** $c \leftarrow \bot$
return $((h', c), \mathsf{int}', m)$

Fig. 6: A generic construction of ad-hoc anonymous communication protocols, aAC

$\underline{\mathcal{B}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{pk}_1)}$
$\mathsf{Ch} \leftarrow 0,\ (\mathsf{st}, \hat{\mathsf{PK}}, \{\mathsf{sk}_i, m_i, \mathsf{rcv}_i, \mathsf{aux}_i\}_{i=0}^1) \leftarrow \mathcal{A}^{\mathsf{Fwd}\mathcal{O}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{pk}_1)$
$\mathsf{Ch} \leftarrow 1,\ b_{\mathcal{B}} \leftarrow \{0,1\}$
$c_{b_{\mathcal{B}}} \leftarrow \mathsf{PKE.Ch}\mathcal{O}(m_{b_{\mathcal{B}}})$ // *i.e.*, $c_{b_{\mathcal{B}}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{b_{\mathcal{C}}}, m_{b_{\mathcal{B}}})$
$c_{1 \oplus b_{\mathcal{B}}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{1 \oplus b_{\mathcal{B}} \oplus b}, m_{1 \oplus b_{\mathcal{B}}})$
$(h_i, \mathsf{int}_i) \leftarrow \mathsf{aOF.Enc}(\mathsf{rcv}_{i \oplus b'}, \mathsf{aux}_i), i \in \{0,1\}$
$p_i := (h_i, c_i), i \in \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{Fwd}\mathcal{O}}(\mathsf{st}, \{(p_i, \mathsf{int}_i)\}_{i=0}^1)$
return b'

$\underline{\mathsf{Fwd}\mathcal{O}(i, \mathsf{PK}, P, \mathsf{aux})}$
parse p **as** (h, c)
if $\mathsf{Ch} = 1$ **then**
  $m \leftarrow \bot$
else
  $m \leftarrow \mathsf{PKE.Dec}\mathcal{O}(i, c)$
endif
$(h', \mathsf{int}') \leftarrow \mathsf{aOF.Fwd}(h, \mathsf{aux})$
return $((h', c), \mathsf{int}', m)$

Fig. 7: Reduction $\mathcal{B}$ in the proof of Theorem 5.4

since the information of the receiver might be present in both the header and the ciphertext.

Formally, suppose $\mathcal{A}$ is a PPT adversary against the receiver-anonymity of aAC. We wish to construct a PPT adversary $\mathcal{B}$ who interacts in the experiment $\mathsf{IK\text{-}CCA1}^{b_{\mathcal{C}}}$ with the key-privacy challenger $\mathcal{C}$ of PKE. $\mathcal{B}$ simulates the $\mathsf{Rcv\text{-}Anon}^b$ experiments for $\mathcal{A}$ as shown in Fig. 7.

$\mathcal{B}$ receives pp, pk_0, and pk_1 from the key-privacy challenger, and is given access to the PKE decryption and challenge oracles $\mathsf{Dec}\mathcal{O}$ and $\mathsf{Ch}\mathcal{O}_{b_{\mathcal{C}}}$ respectively. It forwards $(\mathsf{pp}, \mathsf{pk}_0, \mathsf{pk}_1)$ to $\mathcal{A}$, and simulates the forwarding oracle $\mathsf{Fwd}\mathcal{O}$ for $\mathcal{A}$. The encapsulation oracle is omitted since the encapsulation algorithm is public. To answer the queries $(i, \mathsf{PK}, p, \mathsf{aux})$ to $\mathsf{Fwd}\mathcal{O}$, $\mathcal{B}$ parses $p = (h, c)$, redirects (i, c) to the PKE decryption oracle $\mathsf{Dec}\mathcal{O}$ oracle to obtain m. $\mathcal{B}$ also runs $(h', \mathsf{int}') \leftarrow \mathsf{aOF.Fwd}(h, \mathsf{aux})$ and returns $((h', c), \mathsf{int}', m)$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs $\hat{\mathsf{PK}}$ and $\{m_i, \mathsf{rcv}_i, \mathsf{aux}_i\}_{i=0}^1$.

Since $b_{\mathcal{C}}$ is unknown to $\mathcal{B}$, the latter samples a random bit $b_{\mathcal{B}}$ as a guess of $b_{\mathcal{C}} \oplus b$. In other words, $b_{\mathcal{B}}$ is a guess of which key-privacy game it is playing. $\mathcal{B}$ then queries the PKE challenge oracle $\mathsf{Ch}\mathcal{O}_{b_{\mathcal{C}}}$ on $m_{b_{\mathcal{B}}}$. In return, it receives from the challenge oracle a ciphertext $c_{b_{\mathcal{B}}}$ encrypting $m_{b_{\mathcal{B}}}$ to $\mathsf{pk}_{b_{\mathcal{C}}}$. $\mathcal{B}$ simulates the remaining of the packets as follows. First, it simulates the remaining ciphertext by $c_{1 \oplus b_{\mathcal{B}}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{1 \oplus b_{\mathcal{B}} \oplus b}, m_{1 \oplus b_{\mathcal{B}}})$. Next, it simulates the headers by $(h_i, \mathsf{int}_i) \leftarrow \mathsf{aOF.Enc}(\mathsf{rcv}_{i \oplus b}, \mathsf{aux}_i)$ for $i \in \{0,1\}$. Finally, $\mathcal{B}$ sends $\{(p_i := (h_i, c_i), \mathsf{int}_i)\}_{i=0}^1$ to $\mathcal{A}$.

$\mathcal{B}$ answers the queries to $\mathsf{Fwd}\mathcal{O}$ as before except that it no longer redirects the ciphertext to the PKE decryption oracle $\mathsf{Dec}\mathcal{O}$, but instead set $m := \bot$. The game terminates as the adversary $\mathcal{A}$ outputs a guess b'.

Suppose $b_{\mathcal{B}} \neq b_{\mathcal{C}} \oplus b'$, then the simulation is considered failed, and we do not further analyze. Fortunately, since $b_{\mathcal{B}}$ is uniformly random, we have $b_{\mathcal{B}} = b_{\mathcal{C}} \oplus b$ with probability $\frac{1}{2}$. In this case, $\mathcal{B}$ has simulated the $\mathsf{Rcv\text{-}Anon}^b$ experiment faithfully. (m_i is encrypted under $\mathsf{pk}_{i \oplus b}$ for $i \in \{0,1\}$.)

We analyze the differences between the cases $b \in \{0,1\}$. Let $\chi_{i,j}$ be the distribution of $\mathsf{aOF.Enc}(\mathsf{rcv}_j, \mathsf{aux}_i)$. The first difference is that, (h_0, h_1) is drawn from $(\chi_{0,0}, \chi_{1,1})$ when $b = 0$, and from $(\chi_{1,0}, \chi_{0,1})$ when $b = 1$. By the obliviousness of aOF, $(\chi_{0,0}, \chi_{1,1})$ is identical to $(\chi_{1,0}, \chi_{0,1})$. The two methods of generating h_b are thus identical in the view of $\mathcal{A}$.

The second difference is that, for $i \in \{0,1\}$, c_i is a ciphertext encrypting m_i to receiver i when $b = 0$, and to receiver $1 \oplus i$ when $b = 1$. Suppose $\mathcal{A}$ can distinguish between the two methods of generating c_i with a non-negligible advantage ϵ, that is $\epsilon := |\Pr[\mathsf{Rcv\text{-}Anon}^0_{\mathsf{aAC},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Rcv\text{-}Anon}^1_{\mathsf{aAC},\mathcal{A}}(1^\lambda) = 1]| > \mathsf{negl}(\lambda)$, then we have $|\Pr[\mathsf{IK\text{-}CCA1}^0_{\mathsf{PKE},\mathcal{B}}(1^\lambda) = 1] - \Pr[\mathsf{IK\text{-}CCA1}^1_{\mathsf{PKE},\mathcal{B}}(1^\lambda) = 1]| \geq \frac{\epsilon}{2} > \mathsf{negl}(\lambda)$, breaking the IK-CCA1-security of PKE.

Remark 3. The generic construction can be instantiated, for example, with the (unconditionally) oblivious ad-hoc OF protocols to be constructed in Section 6, and the Cramer-Shoup [15] public-key encryption scheme. Since the latter can be proven IND-CCA1- and IK-CCA1-secure, the instantiated AC protocol can also be proven private, sender- and receiver-anonymous, both under the random self-reducible DDH assumption in the standard model.

5.3 Recasting The DM Protocol

Recall that in the DM protocol [11], a node encrypts its message to the intended receiver using a key-private PKE scheme, and forwards the ciphertext to a random neighboring node. Upon receiving 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 ad-hoc AC protocol constructed from the above generic approach using an ad-hoc

OF protocol DM-aOF = (Enc, Fwd) defined in Fig. 8a. The executions of the DM-aOF.Fwd represent a simple random walk over a graph, at which a packet travels to each neighboring node with equal probability.

Two important parameters for random walk algorithms are hitting time and cover time. *Hitting time* is the maximum of the expected time for traveling from any starting node to any destination). *Cover time* 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 on general graphs, without using any topological information, are both $O(N^3)$ [16].

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

Intuitively, using a simple random walk algorithm and a fixed TTL value, it is less likely for a node in a 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.

6 Generic Construction of Ad-Hoc OF

In this section, we aim to construct OF protocols which exploit the topological information to improve efficiency. Despite the slight difference in the ad-hoc and persistent variants, we state a unified generic construction for both variants defined upon any transition probability matrix and routing tables of the network. When the context is clear, we use OF to denote both aOF and pOF. From this generic construction, we can plug in the transition probability matrices of any random walk algorithms over graphs to obtain a class of OF protocols. For demonstration, we use the simple random walk algorithm and the β-random walk algorithm by Ikeda *et al.* [17] as examples.

We then introduce a convenient representation of the routing paths from a node as a "connectivity matrix" which is computed using partial topological information. Base on the representation, our construction maximizes the minimum probability of successful delivery (p_s) over all potential receivers for each fixed TTL value in one round, *i.e.*, during the transmission from one dummy receiver to another.

6.1 The Idea

Consider a network represented by a strongly connected directed graph. Typically, routing in 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 i chooses a dummy destination 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 j'. When the packet reaches the dummy destination j', node 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 $\mathcal{T} = \{T^k\}_{k=1}^N$. Fig. 8b defines the OF protocols $(P, \mathcal{T})$-(aOF/pOF) (sharing the same code), which is clearly oblivious as the outputs of Enc and Fwd are independent of the receiver rcv.

6.2 Instantiating with 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 DM protocol is a special case of the above generic construction, as stated in Lemma 1.

Lemma 1. Let $P_{\text{sim}} = (p_{kj})_{k,j=1}^N$ and $\mathcal{T}_{\text{sim}} = \{T^k\}_{k=1}^N$ be defined as $p_{kj} = \mathsf{deg}^{-1}(k)$ if $j \in \mathcal{N}(k)$, and $p_{kj} = 0$ otherwise, and $T^k[j] = j \; \forall j$ respectively. Then DM-aOF = $(P_{\text{sim}}, \mathcal{T}_{\text{sim}})$-aOF.

Similarly, we can define a persistent variant of DM as DM-pOF := $(P_{\text{sim}}, \mathcal{T}_{\text{sim}})$-pOF. Alternatively, the transition probabilities may depend on the local topological information, *e.g.*, the degrees of the neighboring nodes. We consider the β-random walk algorithm designed by Ikeda *et al.* [17].

Definition 6.1. The transition probability matrix $P_\beta = (p_{kj})_{k,j=1}^N$ of the β-random walk algorithm is defined as $p_{kj} = \mathsf{deg}^{-\beta}(j)(\sum_{u \in \mathcal{N}(k)} \mathsf{deg}^{-\beta}(u))^{-1}$ if $j \in \mathcal{N}(k)$, and $p_{kj} = 0$ otherwise.

The β-random walk has hitting time and cover time equal to $O(N^2)$ and $O(N^2 \log N)$ respectively on general graphs when $\beta = \frac{1}{2}$ [17]. Thus, β-OF := $(P_\beta, \mathcal{T}_{\text{sim}})$-OF should be asymptotically more efficient than DM-OF.

For 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 no more than T steps. Therefore, we formulate the correctness of $(P, \mathcal{T})$-OF as follows.

Theorem 6.2. 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 $T^k[j] = j \; \forall j$. Then $(P, \mathcal{T})$-OF is (T, ρ)-correct.

6.3 Optimizing p_s

6.3.1 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 will be routed to different destinations. 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.

$\underline{\mathsf{aOF.Enc}(\mathsf{rcv}, \mathsf{aux}_{\mathsf{snd}})}$
parse $\mathsf{aux}_{\mathsf{snd}}$ **as** $\mathcal{N}(\mathsf{snd})$
$\mathsf{int} \leftarrow \mathcal{N}(\mathsf{snd})$
return (L, int)

$\underline{\mathsf{aOF.Fwd}(h, \mathsf{aux}_{\mathsf{int}})}$
parse h **as** ℓ, $\mathsf{aux}_{\mathsf{int}}$ **as** $\mathcal{N}(\mathsf{int})$
if $\ell = 0$ **then return** $(\perp, \perp)$
$\mathsf{int}' \leftarrow \mathcal{N}(\mathsf{int})$
return $(\ell - 1, \mathsf{int}')$

(a) DM-aOF: A construction from DM [11], where $\mathcal{N}(k)$ denotes the set of neighboring nodes of node k

$\underline{\mathsf{aOF.Enc}(\mathsf{rcv}, \mathsf{aux}_{\mathsf{snd}})\ /\ \mathsf{pOF.Enc}(\mathsf{aux}_{\mathsf{snd}})}$
parse $\mathsf{aux}_{\mathsf{snd}}$ **as** $(\mathsf{snd}, (p_{\mathsf{snd},j})_{j=1}^{N}, T^{\mathsf{snd}})$
$\mathsf{rcv}' \leftarrow \chi_{\mathsf{snd}}, \mathsf{int} := T^{\mathsf{snd}}[\mathsf{rcv}']$
return $(h := (\mathsf{rcv}', L), \mathsf{int})$

$\underline{\mathsf{aOF.Fwd}(h, \mathsf{aux}_{\mathsf{int}})\ /\ \mathsf{pOF.Fwd}(h, \mathsf{rcv}, \mathsf{aux}_{\mathsf{int}})}$
parse h **as** (rcv', ℓ), $\mathsf{aux}_{\mathsf{int}}$ **as** $(\mathsf{int}, (p_{\mathsf{int},j})_{j=1}^{N}, T^{\mathsf{int}})$
if $\ell = 0$ **then return** $(\perp, \perp)$
if $\mathsf{rcv}' = \mathsf{int}$ **then** $\mathsf{rcv}' \leftarrow \chi_{\mathsf{int}}$
return $((\mathsf{rcv}', \ell - 1), \mathsf{int}' := T^{\mathsf{int}}[\mathsf{rcv}'])$

(b) $(P, \mathcal{T})$-OF: Our generic constructions (sharing the same code) for transition probability matrix $P = (p_{kj})_{k,j=1}^{N}$, routing tables $\mathcal{T} = \{T^k\}_{k=1}^{N}$, and distribution χ_k over $[N]$ defined by $(p_{kj})_{j=1}^{N}$

Fig. 8: Constructions of OF protocols (L denotes a constant TTL value, which is typically ∞ for $(P, \mathcal{T})$-pOF)

6.3.2 The Construction

We design a routing strategy which is independent of 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 with transition probability $x = (x_i)_{i=1}^{N}$. Let A be the connectivity matrix of the node defined in § 6.3.1. 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

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

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 given by $\hat{x}_i = \frac{1}{|I|}$ if $i \in I$, and $\hat{x}_i = 0$ otherwise, where $I = \{i : \|A_i\|_1 = 1\}$ and A_i is the i-th row of A.

Remark 4. 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 one 1, which is $A(j,j)$ (*i.e.*, node j is a neighboring node).

6.3.3 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 [18, Theorem 4.4], $\hat{x}$ is an optimal solution to (1).

Lemma 2. The optimal solution to

$$\max\ (\min_i\ (Ax)_i) \text{ s.t. } x \geq 0 \wedge \|x\|_1 = 1 \qquad (1)$$

where $x \geq 0$ means $x_i \geq 0\ \forall i \in [N]$, is given by $\hat{x}_i = \frac{1}{|I|}$ if $i \in I$, and $\hat{x}_i = 0$ otherwise, where $I = \{i : \|A_i\|_1 = 1\}$ and A_i is the i-th row of A.

To prove Lemma 2, let $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^N$ and consider the equivalent model of (1) given by

$$\min -p \text{ s.t. } (Ax \geq pe) \wedge (e^T x = 1) \wedge (x \geq 0) \wedge (p \geq 0). \qquad (2)$$

The dual of the primal problem (2) is

$$\max d \text{ s.t. } (A^T y \leq -de) \wedge (e^T y \geq 1) \wedge (y \geq 0) \qquad (3)$$

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

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

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

From 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.*, $d = -p = -\frac{1}{|I|}$. By the LP Strong Duality Theorem [18, Theorem 4.4], $\hat{x}$ is an optimal solution to (2) and to (1).

6.3.4 The Optimized Scheme

Finally, we obtain our optimized schemes Opt-OF $= (P_{\text{opt}}, \mathcal{T}_{\text{opt}})$-OF, where $P_{\text{opt}} = (p_{kj})_{k,j=1}^{N}$, $p_{kj} = \hat{x}_j^k\ \forall k, j$.

It is not easy to formulate the correctness of this construction, at least not in a clean equation form as in the generic construction. The difficulties arise from that the dummy receiver which was 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 6.3. 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-OF is $(l, \frac{1}{|I|})$-correct.

7 Non-Black-Box Multi-Casting aAC

In § 5, we have constructed aAC generically from key-private PKE and aOF, which send messages one at a time. Plugging in the generic construction of aOF in § 6, we obtain an aAC protocol which works as follows: The sender encrypts its message using the PKE, assign a fixed TTL value to the header, then forward the packet to a neighbor according to the underlying random walk algorithm. Naively, this can be extended so that multiple messages are encrypted and then forwarded to multiple neighbors. No efficiency is gained by using this approach, as it is equivalent to calling the encapsulation algorithm multiple times independently.

Consider the following extension of our optimized Opt-aOF. Instead of taking one receiver as input, suppose there are multiple receivers $\{\mathsf{rcv}_j\}_{j=1}^{J}$ to which the sender wishes to encapsulate headers. Naturally, the sender picks J paths to J leaf nodes independently (with or without replacement). To maximize the number of receivers who can successfully receive their messages, the sender rearranges the ordering of the paths, so as to maximize the number of receivers who are located in the assigned paths. In this way, those receivers are guaranteed to receive their messages in the first round, while the others may still have a chance when the second-round dummy receivers are chosen.

Unfortunately, the above extension cannot be used as a black-box to construct the multi-casting aAC since it is not oblivious! This is because the ordering of the output paths depends on the input receivers. A workaround is to construct the multi-casting aAC directly: The encapsulation algorithm encrypts the J messages to their respective receivers using key-private PKE, and assigns them to the J paths as described above. Then, the important step is to forward the packets to their next hops in a random order. This cancels the undesirable effect of the input-dependent ordering of the paths, yet still maintain the assignments of receivers to paths.

8 Experiments

We are interested in measuring the probability p_s of delivering a packet to the intended receiver successfully, and the expected number of hops h traveled by a successfully delivered packet before reaching the receiver. We conduct experiments to investigate p_s and h of $\{\mathsf{DM}, \beta, \mathsf{Opt}\}$-OF, over randomly generated strongly connected directed graphs, with different TTL L. Our experiment covers both aOF and pOF since their constructions (which are shown in Figure 8b) are quite similar. We thus simply use OF to denote both variants. While it is out of the scope of this paper, in § 8.4, we show how one can calculate the expected time needed for a successful delivery, and the expected network capacity consumed, using the values L, p_s, and h. In other words, measuring only p_s and h is sufficiently meaningful.

In our generic construction, packets are routed in the same way as ordinary (non-anonymous) packets according to the routing tables. We therefore do not perform any traffic analysis, for firstly it is out of scope, and secondly the traffic pattern is identical to that generated by the underlying routing protocol according to the routing tables.

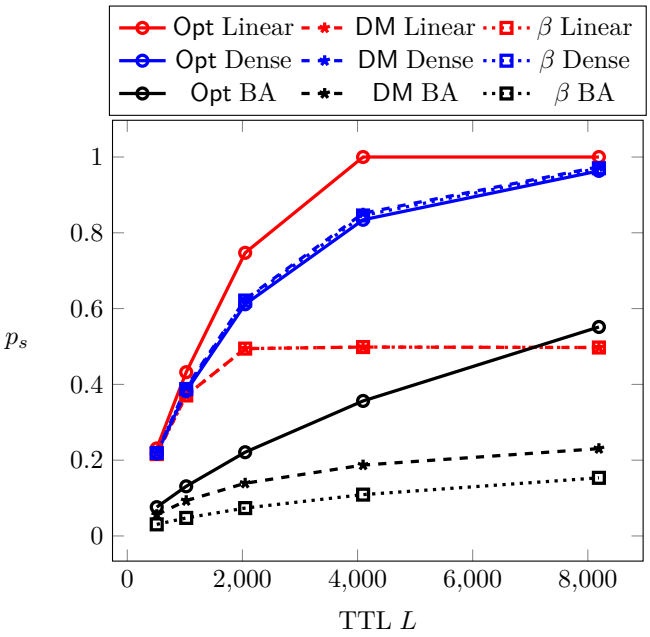

Fig. 9: p_s against TTL

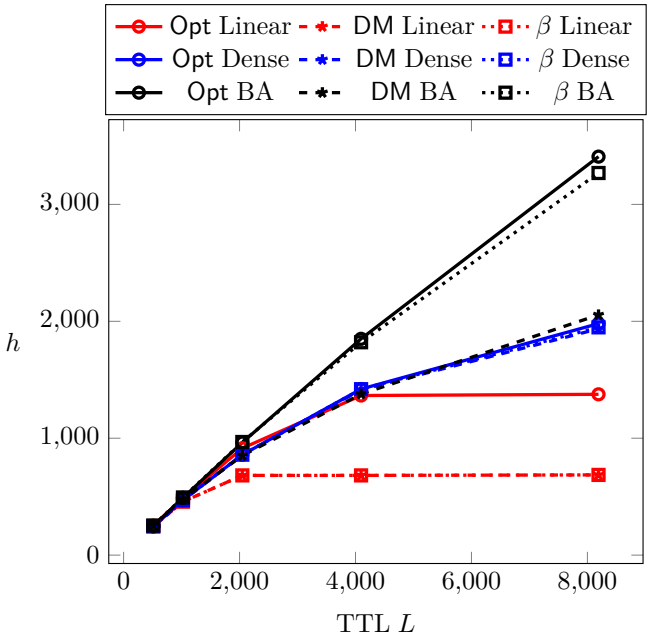

Fig. 10: h against TTL

8.1 Graphs Generation

We repeated our experiments for three types of strongly connected directed graphs with $N = 2048$ nodes.

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* [19]. This model aims to capture the characteristics of real-world networks, which 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*, generated as follows. 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 [11] who set the number of edges M to 32768. This type of graphs is much denser than the graphs representing real world networks.

8.2 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-OF, β-OF, and Opt-OF each for 10 times independently, attempting to deliver a packet to the intended receiver in each instance. The average of p_s and h are calculated for each combination of the graph, protocol, and TTL.

Recall that Opt-OF 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.

8.3 Results

Fig. 9 and 10 respectively show the estimates of p_s and h for different TTL values in each setting. As expected, our Opt-OF design largely outperforms the other two protocols in terms of p_s 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 p_s among the protocols are almost negligible for all TTL values, as the three schemes are almost equivalent in this setting. Our design also has slight disadvantage in terms of the number of hops traveled by the successful packets in most of the cases. This is probably due to the fact that a packet must first travel to a leaf node of a shortest path tree, before the next leaf node is chosen. It is also interesting to see that β-OF has higher hop count than DM-OF in BA graphs, which may be because the former tends to forward packets to less connected neighbors.

It is surprising to see that β-OF performs worse than DM-OF in BA graphs in terms of success probability. 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 $O(N)$. The packets may not live long enough to tell the tale.

Finally, we remark that in the experiment by Young and Yung [11] for their DM protocol, packets are created by all N nodes constantly until the first packet reaches its intended receiver. Therefore, although the average number of hops traveled until successful delivery of their protocol in our dense graphs was measured to be 146 [11], 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. The performance for the nodes located in the more isolated areas might have been neglected.

8.4 Interpretation

For a more reliable communication, suppose that our proposed AC protocol is used with the following simple retransmission strategy. First, the sender includes its identity into the message, so that the receiver knows to whom it should send an acknowledgment (ACK). Next, to raise the probability of successful delivery, the sender sends out packets encapsulating the same message repeatedly at a frequency of f packets per time-step. That is, the sender retransmits every f^{-1} time-steps. For simplicity, we assume that a packet can be forwarded to the next hop in 1 time-step, and all computations can be performed instantly in 0 time-step. Once the message reaches the receiver, the latter sends out acknowledge by encapsulating it in packets of the AC protocol. Same as the sender, the receiver repeatedly sends out packets encapsulating the acknowledgment at a frequency f. The sender stops sending packets encapsulating its message once it receives the acknowledgment from the receiver.

We analyze the efficiency of the above transmission strategy, when the TTL of the packets is set to L. First, we examine the expected delay $d = d(f, L)$ of delivery of a message, *i.e.*, the expected number of time-steps it takes for a message to reach the intended receiver. Since the message takes expected time d to reach the receiver, and the acknowledgment takes another expected time d to reach the sender, the expected time that the sender confirms the successful delivery of the message is $2d$. We now work out the formula for d. Let $p_s = p_s(L)$ be the probability of successful delivery of a packet, and $h = h(L)$ be the expected number of hops traveled by a packet before it is successfully delivered. Note that the delay if the i-th packet is the first to reach the intended receiver is $\frac{i+1}{f} + h$. Thus, denoting $\bar{p_s} = 1 - p_s$, we have

$$\begin{aligned} d &= p_s h + \bar{p_s}(p_s(\frac{1}{f} + h) + \bar{p_s}(p_s(\frac{2}{f} + h) + \bar{p_s}(\ldots))) \\ &= \frac{p_s}{f}\sum_{i=0}^{\infty} \bar{p_s}^i i + p_s h \sum_{i=0}^{\infty} \bar{p_s}^i = \frac{p_s}{f}\left(\frac{1-p_s}{p_s^2}\right) + p_s h \left(\frac{1}{p_s}\right) \\ &= (1-p_s)/(f p_s) + h. \end{aligned}$$

Next, we wish to understand, for a single sender-message-receiver tuple, the expected consumed capacity $c = c(f, L)$,

measured in the maximum expected number of message packets and acknowledgment packets co-existing in the network over the duration of the communication. There are two cases for the expected number of message packets. When $L \leq 2d$, the expected number of message packets increases at a rate of f during time $t \in [0, L]$, stays at a maximum of fL during time $t \in [L, 2d]$, and decreases at a rate of f during time $t \in [2d, 2d + L]$ until it reaches zero. In the second case where $L > 2d$, the expected number increases at a rate of f during time $t \in [0, 2d]$, where at $t = 2d$ the number reaches a maximum of $2fd$, and immediately decreases at a rate of f at time $t \in [2d, 4d]$ until it reaches zero. At time $t = d$, the message is expected to reach the receiver, who starts sending out acknowledgment packets. Thus, during time $t \in [d, d+L]$, the expected number of acknowledgment packets thus increases at a rate of f, until $t = d + L$, where the expected number is capped at fL. To summarize, the expected consumed capacity c is given by $c = f(L + \min(L, 2d))$.

Table 1 shows the efficiency of our schemes in selective settings calculated from the measurements reported in § 8. Specifically, we are interested in the efficiency of the most competitive schemes (Opt-OF and DM-OF) in BA graphs which best reflect real network topologies. We choose to report results for $L = 512$ and $L = 2048$. The former is chosen since the hop count h increases roughly linearly with L, and the expected delay d largely depends on h. The latter is chosen since the distance between any two out of 2048 nodes must be less than 2048. For a meaningful comparison, we tune the frequency f such that the expected consumed capacity per sender-receiver pair per message is either around 100 or around 300. The column Retry shows fd, the expected number of retransmission required for at least one successful delivery. The column $1/f$ shows the time interval that the sender waits until the next retransmission. To summarize, when TTL is small, *e.g.*, $L = 512$, Opt-OF generally outperforms DM-OF. When TTL is large, *e.g.*, $L = 2048$, their efficiencies are comparable.

L	Scheme	Retry	$\frac{1}{f}$	p_s	h	d	c
512	DM	42	10	0.06	256	424	102
512	Opt	37	10	0.08	249	370	102
512	DM	102	3	0.06	256	306	341
512	Opt	95	3	0.08	249	285	341
2048	DM	27	41	0.14	851	1105	100
2048	Opt	27	41	0.22	963	1108	100
2048	DM	72	13	0.14	851	932	301
2048	Opt	72	14	0.22	963	1013	291

TABLE 1: Efficiency of Retransmission Strategy

References

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[2] S. S. M. Chow, W. Susilo, and T. H. Yuen, "Escrowed linkability of ring signatures and its applications," in *Progress in Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, pp. 175–192, 2006.

[3] S. S. M. Chow, J. K. Liu, and D. S. Wong, "Robust receipt-free election system with ballot secrecy and verifiability," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*, 2008.

[4] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pp. 303–320, 2004.

[5] A. Pfitzmann and M. 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, Aug. 2010. v0.34.

[6] A. Hevia and D. Micciancio, "An indistinguishability-based characterization of anonymous channels," in *Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*, pp. 24–43, 2008.

[7] N. Gelernter and A. 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*, pp. 225–236, 2013.

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

[9] A. Hirt, M. J. J. Jr., and C. 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.

[10] A. Hirt, M. J. J. Jr., and C. 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*, pp. 269–278, 2008.

[11] A. L. Young and M. 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*, pp. 157–165, 2014.

[12] M. Bellare, A. Boldyreva, A. Desai, and D. 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*, pp. 566–582, 2001.

[13] P. Mittal, F. G. Olumofin, C. Troncoso, N. Borisov, and I. 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.

[14] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract)," in *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, pp. 325–335, 2004.

[15] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pp. 13–25, 1998.

[16] G. Brightwell and P. Winkler, "Maximum hitting time for random walks on graphs," *Random Struct. Algorithms*, vol. 1, pp. 263–276, Oct. 1990.

[17] S. Ikeda, I. Kubo, and M. Yamashita, "The hitting and cover times of random walks on finite graphs using local degree information," *Theor. Comput. Sci.*, vol. 410, pp. 94–100, Jan. 2009.

[18] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization.* Athena Scientific, 1st ed., 1997.

[19] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.