Formal Notions of Anonymity for Peer-to-peer Networks

Jiejun Kong
Department of Computer Science
University of California, Los Angeles, CA 90095
jkong@cs.ucla.edu

Abstract

Providing anonymity support for peer-to-peer (P2P) overlay networks is critical. Otherwise, potential privacy attacks (e.g., network address traceback) may deter a storage source from providing the needed data. In this paper we use this practical application scenario to verify our observation that network-based anonymity can be modeled as a complexity based cryptographic problem. We show that, if the routing process between senders and recipients can be modeled as abstract entities, network-based anonymity becomes an analogy of cryptography. In particular, perfect anonymity facing an unbounded traffic analyst corresponds to Shannon's perfect secrecy facing an unbounded cryptanalyst. More importantly, in this paper we propose Probabilistic Polynomial Route ($\mathcal{PPR}$) model, which is a new polynomially-bounded anonymity model corresponding to the Probabilistic Polynomial Time ($\mathcal{PPT}$) model in cryptography. Afterwards, network-based anonymity attacks are with no exception in $\mathcal{BPP}$. This phenomenon has not been discovered in previous anonymity research.

Keywords—Formal notion of anonymity; Probabilistic Polynomial Route model; Chosen Recipient Attack; Chosen Sender Attack; Chosen Route Attack; P2P network.

1 Introduction

Fully distributed peer-to-peer (P2P) networks [19][30][26] provide a scalable, intrusion tolerant, and self-organizing substrate for cooperative applications. In such overlays, every node is assigned a unique pseudonym, referred to as a "virtual id", which is chosen from a large anonymity set. Given a message and a destination, these overlays efficiently route the message to the destination. Generally, such overlays maintain $O(n)$ state and provide routing paths of $O(n)$ expected hops, with $O(2^n)$ number of nodes in the network.

Providing anonymity support for the P2P overlay networks is critical. Otherwise, potential privacy attacks (e.g., network address traceback) and subsequent social threats may deter a storage source from providing the needed data. We use this practical application scenario to propose a set of formal notions of network-based anonymity.

Our formal notions transform network-based anonymity into cryptography. We show that, if the routing process between senders and recipients can be modeled as abstract entities, network-based anonymity is an analogy of cryptography. In particular, anonymity is classified into two major categories: the information-theoretic perfect anonymity and complexity-based anonymity. In information-theoretic perfect anonymity, a security solution pays exponential overhead to resist an unbounded traffic analyst. This corresponds to Shannon's perfect cipher which pays exponential overhead (i.e., 2^n messages require 2^n keys) to resist an unbounded cryptanalyst. We believe that such an impractical adversary should be changed to a more practical adversary whose capability is polynomially bounded. Therefore, we propose a set of formal notions based on the Probabilistic Polynomial Route ($\mathcal{PPR}$) model, which is a *new anonymity model* corresponding to the Probabilistic Polynomial Time ($\mathcal{PPT}$) model in cryptography. We further propose a set of advanced anonymity attacks, namely chosen recipient attack (CRA) against sender anonymity, chosen sender attack (CSA) against recipient anonymity, and chosen route attack (CRtA) against both sender and recipient anonymity. Though these advanced anonymity attacks have not been studied in previous research, they are feasible attacks that can be realized by a more capable but polynomially-bounded adversary. In addition to these new notions to formalize anonymity problem statements, we also propose fully distributed polynomially-bounded (in particular quadratically bounded) solutions to ensure $\mathcal{PPR}$ anonymity in a scalable P2P network.

The paper is organized as follows. Section 2 explains background and related work. The information-theoretic perfect anonymity, the complexity-based $\mathcal{PPR}$ model and associated formal notions are specified in Section 3. In Section 4 we present a prototype solution to implement $\mathcal{PPR}$ anonymity using $O(n^2)$ time-complexity and $O(n)$ space-complexity. Finally Section 5 concludes the paper.

2 Background and Related Work

2.1 Informal anonymity notions

Pfitzmann and Köhntopp [16] introduced a set of informal definitions that characterizes anonymity threats in fixed networks. In a distributed system or computer network, members are identified by unique IDs. Network transmissions are treated as the *items of interest* (IOIs). *Pseudonym* is an identifier of subjects to be protected. It could be associated with a sender node, a recipient node, or any protégé demanding protection. The concept of *pseudonymity* is defined as the use of pseudonyms as IDs. The concept of *anonymity* is defined as the state of being not identifiable with a set of subjects, namely the *anonymity set*. In a network comprised of network nodes, the anonymity set is the set of all (uncompromised) network members in the network.

The concept of *anonymity* is defined in terms of *unlinkability* or *unobservability*. The difference between unlinkability and unobservability is whether security protection covers IOIs or not:

- *Unlinkability*: Anonymity in terms of unlinkability is defined as unlinkability of an IOI and a pseudonym. An anonymous IOI is not linkable to any pseudonym, and an anonymous pseudonym is not linkable to any IOI. More specifically, *sender anonymity* means that a particular transmission is not linkable to any sender's pseudonym, and any transmission is not linkable to a particular sender's pseudonym. *Recipient anonymity* is similarly defined.

  A property weaker than these two cases is *relationship anonymity* where two or more identity pseudonyms are unlinkable. In particular for senders and recipients, it is not possible to trace who communicates with whom, though it may be possible to trace who is the sender, or who is the recipient. In other words, sender's pseudonym and recipient's pseudonym (or recipients' pseudonyms in case of multicast) are unlinkable.

- *Unobservability*: Unobservability also protects IOIs from being exposed. That is, the message transmission is not discernible from random noise. More specifically, *sender unobservability* means that a could-be sender's transmission is not noticeable. *Recipient unobservability* means that a could-be recipient's transmission is not noticeable. *Relationship observability* means that it is not noticeable whether anything is sent from a set of could-be senders to a set of could-be recipients.

Unobservability states that a transmission is *not* interceptable by adversary. In a network with finite members[1], this can be achieved either by (1) making network transmissions indistinguishable from random physical noises, or (2) maintaining radio silence. The first method may be useful to fool external adversary. But in the presence of internal adversary, anonymity in terms of unobservability can only be achieved by radio silence (or equivalence) so that nobody can receive it. Throughout the paper, we use the term "anonymity" as a synonym of "anonymity in terms of unlinkability".

2.2 Formal anonymity notions

Information-theoretic models for anonymity were independently proposed in [23] and [9]. Our perfect anonymity model will show that these models can be translated into an equivalent form of Shannon's classic notion of perfect secrecy.

Rackoff and Simon [18] and Berman et al. [1] studied how to build a complexity based infrastructure for anonymity, but both of them are totally different from our approach. In a network of N nodes, Rackoff and Simon's adversary model allows $O(1)$ fraction of compromised nodes and *all* compromised links. Berman et al's adversary model allows $O(1)$ fraction of compromised nodes and $O(1)$ fraction of compromised links. Neither fraction is same as the one used in our $\mathcal{PPR}$ anonymity model. In Rackoff and Simon's model, brute-force transmission at each timestep must be implemented to resist a unbounded timing analyst. This impractical adversary model must be relaxed. But a relaxation to $O(1)$ may be too much. In contrast, we relax the adversary's capability to $O((\log N)^c)$ (where c is a constant characterizing polynomial bound), a quantity that illustrates the straight-forward mapping between cryptography and anonymity. This is not achieved in [18] and [1].

3 Formal Models for Network-based Anonymity

In this section we employ a new approach to model network security (in particular network-based anonymity). The model is equivalent to the indistinguishable security model used in modern cryptography, but no similar model has ever been used in network security research.

3.1 Underlying assumption

A scalable P2P network on the Internet is a virtual overlay where pairwise communication has nearly uniform cost (i.e., all nodes are one logical hop away). Currently, the pairwise end-to-end latency between any two core Internet routers (located in the contiguous 48 states) is less than 100ms, and the pairwise end-to-end latency between any two peer nodes is largely affected by the last-mile connection technology (dialup phone, cable, or DSL). Therefore, the P2P overlay network can be regarded as a virtually fully-connected network with an average pairwise peer-to-peer communication delay τ. Clearly, we can use the overlay network to construct a hypercube network with the same number of vertexes/peers but less number of edges (see Appendix B). In particular, it was shown in [20] that Content-Addressable Network [19] (CAN), a well-known P2P network structure, organizes peer nodes into a hypercube-like structure (Section 4 will show a fully distributed design to organize network peers into a strict hypercube).

[1]Unlike Serjantov's Ph.D. thesis [22], where unobservability is defined on an amorphous network with or without upperbound in anonymity set size and traffic amount, we only consider *finite* anonymity sets and *finite* traffic amount. Therefore, dummy traffic, which is also part of IOI, can ensure unlinkability but not unobservability.

The number of vertexes in the n-hypercube Q_n is $N = 2^n$. If the number of peer nodes in the P2P network, $2^n < N < 2^{n+1}$, cannot form an exact hypercube, then we form a hypercube network with some empty vertexes. All edges connected to the empty vertexes are also removed. For the ease of analysis, we will use n-hypercube Q_n in all discussions. The number of edges in Q_n is $2^{n-1} \cdot n$. For two diagonal nodes (i.e., two nodes with maximal distance) in Q_n, a simple oblivious routing strategy is to choose next stop by flipping one bit. In optimal routing (i.e., with minimal hop count en route) like the one specified in Algorithm 3, a flipped bit will not be flipped again. Therefore, there are totally $n!$ possible routes between the two diagonal nodes. By Stirling's Formula

$$n! = \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(1 + \frac{1}{12n} + O\left(\frac{1}{n^2}\right)\right),$$

the number of possible routes between the two diagonal nodes is an exponential quantity, which is larger than any polynomial when n is sufficiently large. It is easy to ensure this super-polynomial property for routes between any pair of nodes (even in the worst case the pair of nodes are neighbors) by a tractable tradeoff on efficiency—the source sender can always use its diagonal node[2] as a relay.

Like other formal notions, our assumptions may be questioned when applied to the real world.

Remark 1 *It is valid to question our model by saying its practicality is limited by the scale of a P2P network. A P2P network with 2^{30} peer nodes (where the network scale in our is model measured by the network diameter[3] 30) may be possible today, but one with 2^{128} is clearly not.*

We believe the question can be answered in two ways.

1. *First, compromising different peer nodes and tapping different links in a very large scale network (like the Internet) is not as easy as encrypting/decrypting a block of bits on a CPU. Think about a modern crypto-system that cannot be compromised by a polynomially-bounded adversary with non-negligible probability, the reasonable key size of the crypto-system is determined by the per-encryption/decryption cost using state-of-art computation capabilities (as analyzed in [13]). What modern cryptographers have developed is the asymptotic bound between the cost of the adversary and the cost of the legitimate side. When the per-encryption/decryption cost is much higher (as in 1950s), the key size is allowed to be much smaller.*

   Similarly, the scale of a network is limited by the per-node & per-link deployment cost. What we are developing is also the asymptotic bound between the cost of the adversary and the cost of the legitimate side. If the cost of compromising a peer node and tapping a link is high, then the network scale is not necessarily large.

2. *Second, as the paper title says, we are interested in formal notions that will reveal the asymptotic bounds between the cost of the adversary and the cost of the legitimate side. A deployment-wise argument is valid against empirical system design, but should not be used to discourage a formalization effort.*

Remark 2 *It is also valid to raise efficiency concerns about polynomially bounded algorithms. In practice, not all polynomially bounded algorithms are efficient on nowadays hardware. Nevertheless, the new formal model proposed in this paper is merely a foundation of developing new anonymous protocols that are very different from current notion of k-anonymity [27][29]. In a practical anonymous protocol design, we believe that the protocol efficiency can be significantly improved by state-of-art network research, system research and cryptologic research.*

[2] In this paper, the upper half of Q_n is allowed to be empty, thus the source should use its diagonal node in the lower half Q_{n-1}.

[3] Network diameter is defined as the maximal of minimal hop count between any pair of network nodes.

3.2 Information-theoretic perfect anonymity: an analogue to Shannon's perfect secrecy

At first, we use an intuitive argument to demonstrate that perfect anonymity can be achieved for an anonymity set of finite size. The concept of perfect anonymity in information theoretic models is similar to the concept of perfect secrecy proposed in Shannon's information theoretic secrecy paper [25].

Shannon developed the notion of *perfect secrecy* for message encryption based on information theory. Perfect cipher is a mathematical relation amongst three random variables K, M, E in *finite* key space $\mathcal{K}$, message/plaintext space $\mathcal{M}$, and cryptogram/ciphertext space $\mathcal{E}$, respectively. $H(M)$ denotes the entropy of M. $H(M|E)$ denotes the conditional entropy of M after cryptograms are intercepted by an external adversary. The entropy difference $A(M,E) = H(M) - H(M|E)$ is the amount of information about M which an external adversary obtains.

Intuitively, the adversary gains zero information about message in a perfect system even if it intercepts all cryptograms, i.e., $H(M) = H(M|E)$. Given any cryptogram $e \in \mathcal{E}$, Figure 1 states that the adversary has to uniformly choose the candidate message m from the entire message space $\mathcal{M}$ if the secret key k is uniformly distributed over the key space $\mathcal{K}$[4]. Hence each candidate message is equally likely. As a result, although an adversary knows the finite spaces *a priori* and intercepts all cryptograms, information gained *a posteriori* is no more than the *a priori* knowledge. Any adversary cannot compromise perfect message secrecy even if it is given infinite time to exhaustively search the entire finite spaces. It was shown that *one-time pad* (or *Vernam cipher* [28] with one-time key bits) achieves perfect secrecy as long as the number of keys is not less than the number of messages.

Similarly, ideal anonymity can be defined on uniform distributions and the difference between *a priori* and *a posteriori* knowledge. For an anonymity set of size N, if the adversary's *a priori* knowledge (which is quantified by entropy) about the sender/recipient before attack (i.e., intercepting IOI) equals the *a posteriori* knowledge after attack, then we have perfect sender/recipient anonymity.

As depicted in Figure 3, simply by replacing plaintext M, key K, ciphertext E with sender anonymity set S, recipient anonymity set R and end-to-end communication event set X (in this case data delivery is an analogue of encryption, delivery acknowledgement is an analogue of decryption), respectively, we have the upper-bound of anonymity protection, namely *perfect anonymity*, in a system of finite members.

The above intuitive view is presented for the ease of comprehension. We formally specify perfect anonymity below:

Definition 3 ***(Perfect anonymity)*** *For a set of event space S, let X_S be a discrete random variable with probabilistic distribution $p(i) = \mathsf{Pr}[X_S = i]$ where i represents each possible value that X_S can take. If the event space S denotes an anonymity set, then X_S represents the identity pseudonyms. If the event space S denotes the set of all end-to-end communication events, then X_S represents the end-to-end routing path (being eavesdropped) between any sender and any recipient.*

The adversary's a priori knowledge about the sender/recipient is measured by the uncertainty entropy before any communication event occurs:

$$H(X_{AS}) = -\sum_{i \in X_{AS}} p(i) \cdot \log p(i)$$

where AS is the anonymity set.

[4] People normally think perfect cipher is simply keyed exclusive-OR $XOR(key, \cdot)$. In this paper, we use any keyed transformation f_{key}. This conforms to Shannon's original notion, which explicitly states keys are used in transformations. Keyed exclusive-OR is just a specific form of keyed transformation. Indeed, any family of permutations that constructs Latin Square is qualified according to Shannon.

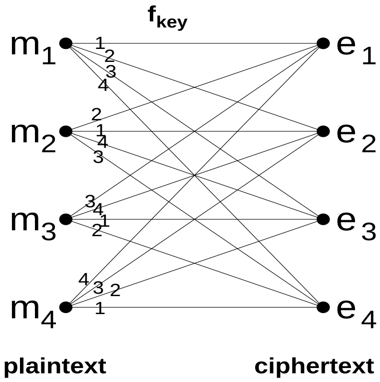

Figure 1: **Perfect cipher** $f_{key}(m_i) \mapsto e_j$ (f_{key}, the keyed transformation functions, are denoted as numbers)

	e_1	e_2	e_3	e_4
m_1	1	2	3	4
m_2	2	1	4	3
m_3	3	4	1	2
m_4	4	3	2	1

Figure 2: **Perfect cipher as Latin Square** (a *de facto* XOR. e.g., 1=00, 2=01, 3=10, 4=11)

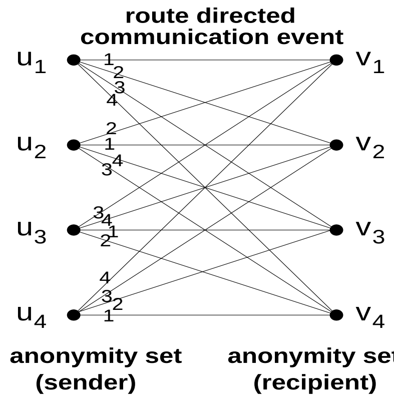

Figure 3: **Perfect anonymity when** $N = 4$ (End-to-end communication events are denoted as numbers)

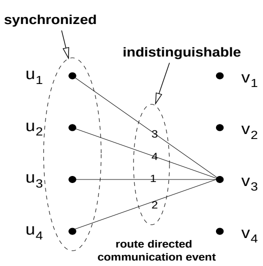

Figure 4: **Perfect sender anonymity: synchronized senders & real events indistinguishable from dummy/decoy events** (using an explicit recipient r_3 in example)

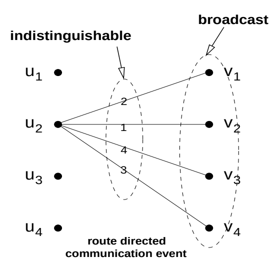

Figure 5: **Perfect recipient anonymity: broadcast to all recipients & real events indistinguishable from dummy/decoy events** (using an explicit sender s_2 in example)

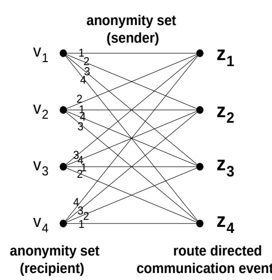

Figure 6: **Sender anonymity** (public communication & chosen recipient)

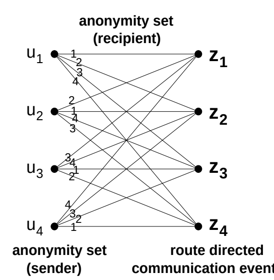

Figure 7: **Recipient anonymity** (public communication & chosen sender)

The adversary's a posteriori knowledge about the sender/recipient is measured by the uncertainty entropy after all communication events occur (which are intercepted by the adversary):

$$H(X_{AS}|C) = -\sum_{i \in X_{AS}, j \in C} p(i,j) \cdot \log\, p(i|j) \tag{1}$$

where AS is the anonymity set, C is the set of intercepted communication events, and conditional probability $p(i|j) = \frac{p(i,j)}{\sum_{i \in X_{AS}} p(i,j)}$.

An anonymous communication scheme ensures perfect anonymity *for sender or recipient if $H(X_{AS}) = H(X_{AS}|C)$. Otherwise the compromise of anonymity is measured by the difference or the ratio between these two quantities.* ■

Remark 4 *Equation (1) is the simplified result. It is derived from the procedure below.*

Suppose X_{AS} and X_C are two random variables on sample space sets AS and C, respectively. For any fixed value j of X_C, by the definition of entropy we have

$$H(X_{AS}|j) = -\sum_{i \in X_{AS}} p(i|j) \cdot \log\, p(i|j).$$

Thus the conditional entropy $H(X_{AS}|X_C)$ is the weighted average of the entropies $H(X_{AS}|j)$ over the probability distribution of j

$$H(X_{AS}|X_C) = -\sum_{i \in X_{AS}, j \in X_C} p(j) \cdot p(i|j) \cdot \log\, p(i|j) = -\sum_{i \in X_{AS}, j \in X_C} p(i,j) \cdot \log\, p(i|j).$$

The conditional entropy on the entire set C is a special case when the random variable X_C takes on C

$$H(X_{AS}|C) = -\sum_{i \in X_{AS}, j \in C} p(i,j) \cdot \log\, p(i|j).$$

Example 5 *Let's consider a random network of four nodes $AS = \{v_1, v_2, v_3, v_4\}$. Suppose during the entire network lifetime, the only communication event is a unicast from v_1 to v_3, and the packets are all in cleartext. The adversary's a priori knowledge about the sender and the recipient is $H(X_{AS}) = 2$. The adversary's a posteriori knowledge about the sender and the recipient is $H(X_{AS}|C) = 0$ after its timing analysis and content analysis.*

Example 6 *Suppose during the entire network lifetime, in the same network the only communication event is a multicast from v_1 to $\{v_2, v_3\}$, and the packets are all in cleartext. The adversary's a priori knowledge about the sender and the recipient is $H(X_{AS}) = 2$. The adversary's a posteriori knowledge about the sender is $H(X_{AS}|C) = 0$, and about the recipient is $H(X_{AS}|C) = 1$.*

[DC-net: complete graph] Figure 3 explains the reason why Chaum's DC-net [5] ensures perfect anonymity. Let's enumerate the conditions to implement perfect anonymity by DC-net:

- The system lifetime is divided into t unit time slots. In [5], only one message communication is discussed, thus $t = 1$.
- Every pair of nodes share an $l \cdot t$-bit key *a priori*. In [5], only one bit communication (i.e., whether to pay the bill for dining cryptographers) is discussed, $l = 1$. However, we must add a 1-bit decoy flag to differentiate real transmissions from decoy transmissions, then $l = 2$.

- The DC-net topology is a fully connected network (the closed ring topology suggested in [5] is discussed right below).
- Every node must synchronously send out a packet (of a uniform packet format) to all immediate neighbors (i.e., all other nodes) per unit time slot. The ith l-bit key is used at the ith time slot. To transmit a perfectly anonymous message of $l-1$ bits, the real sender assembles a packet with the decoy flag *unset* and enciphers the entire packet by Vernam cipher using the l-bit key shared with the real recipient, while each decoy packet *sets* its decoy flag and holds $l-1$ truly random bits.

The DC-net of complete graph cannot deliver more than $(l-1)\cdot t$ bits perfectly anonymous message for all communicating pairs, but it is not vulnerable to an adversary with unbounded computation resource and unbounded network resource.

[DC-net: closed ring] If the DC-net topology is the closed ring as suggested in [5], then it emulates a fully connected network by using other nodes as forwarders:

- The unit time slot must be longer than the round-trip forwarding delay on the ring. Each link's capacity is large enough (i.e., no congestion happens).
- The number of nodes in the network/ring is not greater than 2^n. Each packet has two n-bit fields: one for the sender and the other for the recipent, and an $(l-n)$-bit payload field. The uniform packet length is $n+l$ bits. The packet format is:

	← Enciphered with l-bit key shared with the recipient	→
sender field n bits	recipient field n bits	message payload $l-n$ bits

  A sender puts its own label in the sender field, puts the real recipient's label in the recipient field (or its own label in the recipient field if this is a decoy packet), enciphers the recipient and the message payload fields using the l-bit key shared with the recipient.

  Note that there is a subtle assumption made in the fully-connected DC-net—a node knows it is the real recipient if it receives an incoming non-decoy packet. This assumption eliminates the need of identifying the sender and the recipient in packet format definition. However, now the assumption is *not* true as packet forwarding is introduced in DC-ring, we have to add the n-bit sender field and the n-bit recipient field. Key management is explained right below. In order to avoid deciphering collisions on the n-bit recipient field, we have to enforce the rule that *distinct* n-bit keys are shared between the sender and different nodes. Moreover, the sender field must be in cleartext so that each receiving node knows which key to use in its deciphering trial.
- Every pair of nodes share an $l\cdot t$-bit key *a priori*. The ith l-bit key is used at the ith time slot.

  For each time slot, the first n bits of the l-bit key bits follow permutation (i.e., a row in Figure 2) in key sharing. For example, in a network of four nodes $\{v_1 = 00, v_2 = 01, v_3 = 10, v_4 = 11\}$, each node may choose an arbitrary row of Figure 2. For the node who has chosen a row, say the second row $(01, 00, 11, 10)$, it randomly shares one value in the row with a node in the network (including itself), e.g., sharing 01 with v_3, 00 with v_2, 11 with v_4 and 10 with v_1 as long as no collision happens in the n-bit key sharing. The other $(l-n)$ key bits are random bits.
- A packet is suppressed by the sender at the time of seeing the packet again.

- For each forwarder, it tries to decipher the recipient and the payload fields using the l-bit key shared with the sender. If the deciphered recipient field matches the node's own label, then the forwarding node knows that it is the real recipient and accepts the deciphered payload. But it will forward the original packet to the next hop as if nothing happened. This way, though the sender field in a packet is public, sender anonymity is achieved because the packet could simply be a decoy (so the "sender" is not really a sender). Recipient anonymity is achieved because each packet travels the entire ring once, a *de facto* network-wide flooding.

  Proof (sketch): We use Figure 5 and Figure 4. In the figures a route-directed communication event between any two nodes u and v is now implemented by multi-hop forwarding from u to v.

  For any sender u, let z denote the enciphered recipient field of its packet, then $z = v \oplus k_{uv}$ for the recipient v ($v = u$ for decoy packets) and k_{uv} is the first n key bits shared between u and v during the current time slot. A packet travels through the entire ring, i.e., it is flooded in the network. This realizes Figure 5 or perfect recipient anonymity. On the other hand, every node must send out a (real or decoy) packet per unit time slot. Hence each node will receive one packet from all other nodes during the current time slot. This realizes Figure 4 or perfect sender anonymity.

The original DC-net paper [5] claimed that this closed ring DC-net is vulnerable to internal attacks. But if perfect anonymity is only defined on the anonymity set comprised of all *uncompromised* nodes, then our study shows that this closed ring DC-net is *not* vulnerable to internal attacks as long as the compromised nodes are protocol compliant (i.e., they forward packets as usual and do not modify packets).

[DC-net: hypercube] Hypercube is another topological structure that ensures perfect anonymity against an unbounded adversary. The design is very similar to closed ring DC-net except the following difference:

- The unit time slot must be longer than the forwarding delay on any n-hop routing path.
- Each packet is flooded over the entire network. An feasible flooding technique is to forward a packet (to all neighbors except the incoming link) only for the first time, and then ignore the same packet later.

3.3 $\mathcal{PPR}$ model

In perfect security, the adversary has available exponential computing resources, thus it does not matter how the keyed transformation f_{key} is implemented—even if the keyed transformation is implemented by a hard $\mathbb{NP}$ problem, the adversary can break it whatsoever. Modern cryptography abandons this assumption, and assumes instead that the adversary is a probabilistic algorithm who runs in polynomial time. We speak of the *infeasibility* of breaking the security system rather than the information-theoretic notion of *impossibility* of breaking the same system.

We believe that network security should be addressed by a similar formal approach. The overlay network monitoring schemes have admitted that only a polynomially many routes can be effectively monitored [6]. Thus we adopt a probabilistic polynomial-space approach. In our security model, *the adversary can control any single route (of polynomial hops) of its probabilistic choice*, and it can control *polynomially many* such routes during its attack lifetime. Nevertheless, during its attack lifetime the adversary cannot successfully break a set of routes if the set's cardinality is a super-polynomial quantity, that is, asymptotically larger than any polynomial (e.g., the entire $2^{n-1} \cdot n$ edge set of a hypercube for all sufficiently large network diameter n's).

Definition 7 ***(Probabilistic Polynomial Route $\mathcal{PPR}$ network model)*** *A* Probabilistic Polynomial Route *($\mathcal{PPR}$) is a polynomially-bounded relation between nodes in the sender anonymity set and nodes in the recipient anonymity set. In a $\mathcal{PPR}$ network, every node is of $O(n^c)$ space-complexity (i.e., maintain $O(n^c)$ state), and every protocol is accomplished in $O(n^c)$ time-complexity (i.e., in $O(n^c)$ steps/hops). In particular for routing, given any sender node as the input, each routing step is treated as a step in a virtual probabilistic polynomial-time Turing Machine (who treats the entire network as a graph), such that the machine always halts at a recipient node after $O(n^c)$ steps using $O(n^c)$ number of coin-flips for the network diameter n and some integer c.* ■

A $\mathcal{PPR}$ adversary can successfully control $O(n^c)$ paths (including all $O(n^c)$ nodes and edges on every path) at its probabilistic selection, but cannot control a super-polynomial number (e.g., exponential number) of paths in the network. A $\mathcal{PPR}$ adversary meanwhile is also a probabilistic polynomial-time ($\mathcal{PPT}$) adversary defined in cryptography. In other words, the adversary *cannot* invert one-way functions, or differentiate cryptographically strong pseudorandom bits from truly random bits in polynomial time, with non-negligible probability.

Our formal security specification is built on top of the identical concepts used in the foundations of modern cryptography. In particular, an anonymous system is secure if any $\mathcal{PPR}$ adversary cannot trace a real route with more than negligible probability in $\mathcal{BPP}$.

Definition 8 *($\mathcal{PPR}$ **anonymity**): Communication is $\mathcal{PPR}$* anonymous *in a network of 2^n nodes if the following three conditions hold:*

1. ***Efficient to maintain**: Z is the routing network. Each node in Z runs a routing algorithm Y in $O(n^c)$ time using $O(n^c)$ space, and each packet in Z is of size $O(n^c)$.*
2. ***Efficient to route**: There exists a route producer algorithm X such that on two inputs: (1) the recipient v and (2) a random string r of length $O(n^c)$ for some integer c, $X(v, r)$ is a* route directive *produced by X in $O(n^c)$ time using $O(n^c)$ space. And $Z_Y(X(v, r))$ means the selected forwarding nodes in the network Z use routing algorithm Y to route the packet to v in $O(n^c)$ steps.*
3. ***Negligible probability to trace**: For every PPR routing algorithm Y', every positive integer c, and all sufficiently large n's,*
$$Pr[Z_{Y'}(X(v, r)) \in Z_Y(X(v, r))] < \frac{1}{n^c}$$
where the probability is taken over the coin-flips of Y' and r. ■

The network diameter n in $\mathcal{PPR}$ anonymity plays the role of key length in $\mathcal{PPT}$ cryptography. Intuitively for IPv4, the quantity $max(n)$ is 32 (there are at most 2^{32} peer nodes in the network); for IPv6, the quantity $max(n)$ is 128; and theoretically n is the linearly increasing network scale.

[Route-directive and onion] In Q_n, for a pair of sender u and recipient v, the random variable r lets u choose a specific path to v. Here the *route-directive bit-vector*[5] $z = u \oplus v$ for any sender u and recipient v. Route is directed by randomly flipping one's in z: if the ith bit of z is 1, then resets the ith bit and goes to the neighbor node whose ith bit is the complement of the current node, until all bits of z are reset. Therefore, if there are d one's in z, then there are $d!$ possible routes between u and v, the random r is used to choose the exact *order* of flipping the one's bits. This uniquely selects a specific route from the $d!$ route pool. For an n-bit z, a straight forward form of r is an $n\lceil\log(n+1)\rceil$-bit random value comprised of n sequenced

[5]The simple notations used in this paper are described in Appendix A.

fields, each field is of $\lceil \log(n+1) \rceil$ bits to denote a bit position in z to be flipped. For example, for a 4-bit z=1010, an instance of r can be $3, 1, -1, -1$ which says at first flipping the most significant bit (bit 3) then bit 1. Each field is represented by a $\lceil \log(n+1) \rceil$-bit value.

Like previously discussed, if we want to ensure that there are exponentially many routes between u and v (i.e., when u and v are neighbors, u must use its diagonal node as relay, otherwise there is only $1!$ possible routes between them), then routing can use the diagonal node in the lower half of Q_n as a relay. In this case, $z = z_1 || z_2$ is a $2n$-bit string comprised of two parts:

1. An n-bit long string $z_1 = u\&(2^{n-1} - 1) \oplus (2^{n-1} - 1)$, which directs routing from u to $u \oplus z_1$.
2. An n-bit long string $z_2 = u \oplus z_1 \oplus v$, which directs routing from $u \oplus z_1$ to v.

By this relaying strategy, there are always $x!(n-1 \leq x \leq 2n)$ possible routes between any pair of u and v.

Now that we have the route-directive $X(v, r)$. The next thing is borrowing a protection from MIX-net [4]. That is, the sender must encode the route-directive $X(v, r)$ using a layered onion structure, so that the adversary cannot see the route by seeing v and r.

Remark 9 *The $\mathcal{PPR}$ anonymity model implies that a viable solution must be* ***fully distributed****. Any countermeasure, such as centralized service or threshold cryptography (where the security protection can only tolerate up to a constant threshold number t of adversarial nodes) that is vulnerable to a polynomial number of adversarial nodes or links, is broken by the definition. In particular for onion encryption, if we don't want to use exponential overhead to acquire 2^n public keys, then we need a fully distributed identity-based encryption (IBE) scheme [24]. Here the term "fully distributed" means the IBE cryptosystem should not have a centralized or polynomially shared authority as specified in [3][24]. A fully distributed IBE is unfortunately not available at the moment when this paper is written.*

[Exemplary design: IBE onion routing as Y] If a fully distributed IBE cryptosystem is feasible, then a sample onion routing protocol can use a uniform packet format described below:

$\leftarrow$ x onion layers (n bits per layer) $\rightarrow$					
outmost layer n bits	2nd outmost layer n bits	$\cdots$ n bits per layer	innermost layer n bits	message payload l bits	random padding $(2n - x) \cdot n$ bits

Here l is the uniform payload length. Packets in the network are of a uniform length of $(2n^2 + l)$ bits.

As described above, for any pair of sender and recipient, it is guaranteed that there are $x(n-1 \leq x \leq 2n)$ hops between them. Therefore, the first $x-1$ fields are real, but the last field is $\perp$. For the ease of presentation, let's denote the real forwarding vertexes as a sequence $\langle v_1, v_2, \cdots, v_x \rangle$ where v_x is the recipient vertex.

The sender sequentially puts $\langle v_2, \cdots, v_{x-1}, v_x \rangle$ in the first $x-1$ fields, and the last field is filled with a random vertex that is *not* a neighbor of v_x. This way, the x-th field is actually $\perp$, an un-routable symbol. The recipient v_x knows it is the recipient once it sees $\perp$.

Now the sender produces the route-directive onion in the following manner. Using the IBE cryptosystem, the sender uses v_x's vertex ID to encrypt[6] the xth field and everything after it (i.e., totally $(2n - x + 1) \cdot n + l$

[6] In practice, encryption is done by XORing the plaintexts with a cryptographically strong pseudorandom ensemble [11]. Given a well-known one-way function f defined on key size n_f, node u can deliver an encrypted message to node v: (1) u selects an n_f-bit random seed s; (2) u encrypts the seed using node ID v (i.e., v's public key); (3) Following Blum-Micali's method [2], u produces a cryptographically strong pseudorandom ensemble using the seed s; (4) u XORes the cleartext with the pseudorandom ensemble. At v's side, it first decrypts the seed s using its private key, then produces the same pseudorandom ensemble and successfully decrypts the contents by XOR. In this case, each onion layer is an $(n_f + n)$-bit field with n_f bits dedicated to store the seed. The granularity of bitwise shift operations is hence $n_f + n$ rather than n.

bits), then uses v_{x-1}'s vertex ID to encrypt the $(x-1)$th field and everything after it (i.e., totally $(2n-x+2)\cdot n+l$ bits), until the entire packet is encrypted with v_1's vertex ID. The result of the layered encryption is the route directive onion $X(v_x, r)$.

At each routing stop, the vertex strips off one layer of the onion:

1. It at first decrypts the onion packet with its own IBE private key;
2. Then it knows the next stop by inspecting the first n bits of the decrypted packet;
3. Then it bitwisely shifts the entire packet to the left for n bits, and appends n truly random bits to fill in the last n bits;
4. The result is forwarded to the next stop.

At the real recipient, it sees an un-routable vertex after decryption, then it accepts the l-bit message payload.

3.3.1 Advanced anonymity attacks

The adversary defined so far is a purely passive eavesdropper and traffic analyst, though it can unconditionally compromise polynomially many routes which are chosen randomly. This corresponds to the basic $\mathcal{PPT}$ adversary in cryptography. Anonymity attacks that are analogous to Chosen Plaintext Attack (CPA) [12], Chosen Ciphertext Attack Type 1 (CCA1) [15] and Chosen Ciphertext Attack Type 2 (CCA2) [17] are also practical. In this section we elaborate on these advanced attacks.

As Shannon pointed out [25], perfect cipher is a symmetric structure where the three dimensions (plaintext m, ciphertext e and keyed transformation f_{key}) can be pairwisely switched. Nevertheless, in real applications the roles of the three dimensions are different and asymmetric.

- Transmitted ciphertexts are public, and *chosen ciphertext attack* (CCA) (on a polynomial number of ciphertexts that are not the target ciphertext) is feasible in cryptography. A *decryption oracle* is supposed to help a CCA attacker obtain corresponding plaintexts.
- Any key must not be revealed (also known as the "Kerckhoff Desiderata"). The goal of cryptanalysis is to reveal the secret key.
- Polynomially-bounded *chosen plaintext attack* is feasible in cryptography.

Similarly, our notion of anonymity should possess similar properties, which are illustrated in Figure 6 and 7. In Figure 6, the symmetric structure is transformed to an equivalent form. This figure shows how to build an analogy between sender anonymity and cryptography.

- Route-directive onions are transmitted in public, and *chosen route attack* (on a polynomial number of routes that are not the target route) is feasible in $\mathcal{PPR}$ sender anonymity. The decryption oracle used in CCA attacks is replaced with a *route trace oracle* that reveals the real recipient upon a given public route-directive onion (even though the onion is protected by certain cryptosystem).
- Any sender must not be revealed. The goal of sender anonymity attack is to reveal the secret sender.
- Polynomially-bounded *chosen recipient attack* is feasible in $\mathcal{PPR}$ anonymity model by definition.

In a nutshell, for sender anonymity, chosen plaintext attack (CPA) in cryptography is transformed into chosen recipient attack (CRA), and chosen ciphertext attack (CCA) is transformed into chosen route attack (CRtA). Both attacks have "adaptive" variants. In other words, given the dynamic output of previous-round chosen route/recipient attack, the adversary can choose the next-round route/recipient according to its optimal choice. A practical example of adaptive chosen route attack is the "tagging attack" proposed by

Danezis et al. [8], where an adversarial node en route modifies a message by embedding a pattern so that the pattern can be recognized later by another adversarial node en route. This effectively implements the "route trace oracle". The adversary's probabilistic choices are adaptively improved by the information from the oracle.

Similar to sender anonymity, Figure 7 shows an analogy between recipient anonymity and cryptography. It is identical to Figure 7 except the roles of sender and recipient are switched, and consequently "chosen recipient attack" (CRA) against sender anonymity is changed to chosen sender attack (CSA) against recipient anonymity.

Remark 10 *In Definition 7, we did not include onion in the definition specification. Now we make it up since the advanced attacks have been described.*

If the adversary compromises polynomial number of routes (including every nodes and edges en route) ***following the uniform distribution****, then a polynomially-bounded hypercube network maintenance protocol specified in Section 4 (i.e., this one is without any cryptographic protection like the IBE onion routing) already trivially satisfies Definition 7. However, the adversary can compromise polynomial number of routes following its* own *probabilistic distributions using its own coin-flips. This distribution is no longer necessarily the uniform distribution. This makes CSA, CRA, CRtA attacks feasible. For example, in Q_n, if the adversary controls a cyclic path and partitions the network into at least two sub-nets: one sub-nets has $O(n^c)$ nodes, and the other sub-net(s) has $O(2^n - n^c)$ nodes, then the system is broken by CRA or CSA because the adversary now knows where is the sender/recipient with non-negligible probability.*

A mistake is to devise a network-based countermeasure like this: every node periodically and randomly selects another vertex label, then switch labels with the node with that label. This proactive design effectively enforces the adversary make random choices following the uniform distribution in node/link compromise. Unfortunately, by inspecting the overhead we can see the overhead is exponential—the overhead of this design is equivalent to a flooding in Q_n after n rounds of network-wide proactive exchange. Thus proactive designs are disqualified because they incur exponential overhead!

This is the reason why cryptographic onion is introduced. Despite the "smart" adversary can compromise polynomial number of ***any*** *set of routes (with every nodes and links en route), the onion localizes the damage to the compromised nodes, who cannot see the (route-directive) messages intended for uncompromised nodes.*

3.3.2 Definitions using indistinguishability

The advanced attacks call for re-defining the concept of $\mathcal{PPR}$ anonymity using indistinguishability (similar to semantic security [12] and non-malleable security [10]).

Definition 11 *($\mathcal{IND} - CRA$* ***sender anonymity****):* *An anonymous network ensures $\mathcal{IND} - CRA$* sender anonymity *with following components:*

1. *Z, the onion routing network[7]. Each node in the onion routing network runs a routing algorithm in $O(n^c)$ time using $O(n^c)$ space, and each packet is of size $O(n^c)$.*
2. *X, the route-directive producer function, gets two inputs: (1) a pair of arbitrary nodes $V = (v_i, v_j)$, where the first one is the real recipient and the second one is a decoy, and (2) a random string r of length $O(n^c)$ for some integer c. $X(V, r)$ produces the route-directive onion z in $O(n^c)$ time using $O(n^c)$ space.*

[7] This is a general notion, not the practical "Onion Routing" protocol [21].

3. Y, *the routing function, gets the onion* z. $Z_Y(z)$ *means the network* Z *employs* Y *to deliver packet to the real recipient* v_i *in* $O(n^c)$ *stops.*
4. *For any* $v \in V$, *for all* $r \in \{0,1\}^{n^c}$, $Y(X(v,r)) = v$.
5. *The system has the property of indistinguishability: for all* $\mathcal{PPR}$ *adversary* Y', *for all integer* c, *and for all sufficiently large* n*'s*

$$\mathit{Pr}[Z_{Y'}(X(v_i,r)) = v_i] \;-\; \mathit{Pr}[Z_{Y'}(X(v_j,r)) = v_i] \;<\; \frac{1}{n^c}$$

where the probability is taken over the internal coin-flips of Y' *and the choice of* r. ∎

Remark 12 *Recall the onion in* $\mathcal{PPR}$ *anonymity is an analogue of ciphertext in* $\mathcal{PPT}$ *cryptography (actually it is an IBE ciphertext). And the recipient is an analogue of plaintext. The adversary* Y' *gains negligible advantage by choosing recipients.*

Definition 13 *(*$\mathcal{IND}-CRtA$ ***sender anonymity****): An anonymous network ensures* $\mathcal{IND}-CRtA$ sender anonymity *with following components:*

1. Z, *the onion routing network. Each node in the onion routing network runs a routing algorithm in* $O(n^c)$ *time using* $O(n^c)$ *space, and each packet is of size* $O(n^c)$.
2. X, *the route-directive producer function, get three inputs: (1) a randomly chosen relay node* w, *(2) the recipient* v *and (3) a random string* r *of length* $O(n^c)$ *for some integer* c. $X(w,v,r)$ *produces the route-directive onion* z *in* $O(n^c)$ *time using* $O(n^c)$ *space.*
3. Y, *the routing function, gets the route-directive onion* z. $Z_Y(z)$ *means* Z *employs* Y *to deliver packet at first to the relay* w, *then to the recipient* v *(both in* $O(n^c)$ *steps). Let's denote* $Z_Y(z) = (w,v)$.
4. *For any* w, v, *for all* $r \in \{0,1\}^{n^c}$, $Z_Y(X(w,v,r)) = (w,v)$.
5. *The system has the property of indistinguishability: for all* $\mathcal{PPR}$ *routing algorithm* Y, *for all integer* c, *and for all sufficiently large* n*'s, there exists* Y' *which is the simulator of* Y *such that*

$$\mathit{Pr}[Z_Y(X(w,v,r)) = (w,v)] \;-\; \mathit{Pr}[Z_{Y'}(X(w',v,r)) = (w,v)] \;<\; \frac{1}{n^c}$$

where w' *in the simulator's testing onion* $X(w',v,r)$ *must be different from* w *in the real onion* $X(w,v,r)$, *and the probability is taken over the choice of* w *and* w', *the internal coin-flips of* Y' *and the choice of* r. ∎

Remark 14 *In the pre-processing mode,* Y' *is only allowed to produce a polynomial size vector* $[X(w_i,v,r)]$, $w_i \neq w$ *for testing before* Y*'s onion* $X(w,v,r)$ *becomes public. In the post-processing mode,* Y' *is allowed to produce the polynomial size vector* $[X(w_i,v,r)], w_i \neq w$ *for testing before and after* Y*'s onion* $X(w,v,r)$ *becomes public. This difference in modes is not significant as the above definition is applicable to both cases.*

The notion (w,v) *is the relation* $R(w,v)$ *used in Dolev-Dwork-Naor paper [10]. We assumed that the network is an undirected graph. Thus a routing algorithm* Y *that connects* w *and* v *actually captures a relation* R *between them. So in our notion* R *is not indispensable.*

Here the choice of w *follows a probabilistic distribution chosen by* Y *(like the distribution* $\mathcal{M}$ *on cleartext messages chosen by DDN's adversary* $\mathcal{A}$*).* $X(w',v,r)$ *is the "molded" version of* $X(w,v,r)$, *which is unknown to the simulator. Like non-malleable cryptography, this definition states that knowing the route-directive onion (which is the analogue of ciphertext) does not help the adversary to gain non-negligible advantage, except we have simplified DDN's notions to fit in the new context.*

The $\mathcal{IND} - CSA$ and $\mathcal{IND} - CRtA$ recipient anonymity are similarly defined except the roles of sender and recipient are switched. Note that we model the underlying network as an undirected graph. Thus whether a node is sender or recipient is really not significant with respect to routing, except the sender is the one who initiates an IOI (communication event).

If a hypercube network N that can be constructed and maintained in $O(n^c)$ time-complexity and $O(n^c)$ space-complexity, then N can be used with a semantically secure IBE cryptosystem (or a non-malleable IBE cryptosystem) to ensure $\mathcal{IND} - CRA$/$\mathcal{IND} - CSA$ anonymity (or $\mathcal{IND} - CRtA$ anonymity). We then study the feasibility of constructing the hypercube network with polynomially bounded overheads.

4 Fully distributed hypercube maintenance

This section studies the feasibility of creating and maintaining a fully distributed hypercube network using $O(n^c)$ steps and incurring $O(n^c)$ space per node. The presented algorithms have been implemented.

It was shown in [20] that Content-Addressable Network [19] (CAN), a well-known P2P network structure, organizes peer nodes into hypercube when a collision free hash function can evenly distribute nodes on a d-torus space. In a network of $O(2^n)$ nodes, each node only maintain state about their $O(n)$ neighbors (and if a neighbor is occupied, the node has a connection with the neighbor). Nonetheless, we use a slightly different algorithm without calling hash functions. In theory, the hash function used in CAN should be a Collision-Resistant Hash Function (CRHF) [7], but in practice a Universal One-Way Hash Function (UOWHF) that can be constructed based on any one-way permutation [14] is used. Unfortunately, in both cases, the probability of collision is negligible, but not zero. Whenever collision happens, the network is no longer a hypercube (especially when the number of vertexes is exactly 2^n). In this section, we show that *fully distributed* hypercube construction can be constructed and maintained with *polynomial* overheads and *without* depending on other mathematical assumptions, such as collision-resistance of CRHF and UOWHF.

As specified in Algorithm 2, a fully distributed hypercube network can be inductively constructed by maintaining a virtual counter. If the counter's value is cnt, then the new node will occupy the vertex whose label is cnt. However, in a fully distributed P2P network, the counter must not exist physically. Otherwise, either there exists single point (thus polynomial number) of compromise in a centralized storage design, or updating the counter incurs exponential overhead in a fully distributed storage design. Algorithm 2 satisfies the constraint using $O(n^2)$ complexity.

Algorithm 1 DIAMETER: Get current network diameter n

Require: No global view of the network is allowed. The algorithm is initiated on any occupied vertex V'.

```
for (n := 0; n < max(n); n := n + 1) do
    Use Algorithm 3 to determine whether vertex 2^{n+1} is occupied.
    if (Vertex 2^{n+1} is not occupied) then
        Break the loop.
    end if
end for
```

Ensure: n is the current network diameter.

The LEAVING algorithm is trivial, as the leaving node issues a faked join to know the lastly occupied vertex, switches the label with the last vertex, then deletes the last vertex (i.e., shut down the connections

Algorithm 2 $JOIN(prefix, n)$: Joining the current network of diameter n

Require: Initiated by any occupied vertex V' known to the joining peer. The parameter $prefix$ is 0 (i.e., empty prefix) at the time of invocation, and Q_0 must exist *a priori*. In other words, the algorithm should be invoked as "$JOIN(0, DIAMETER())$" by V', and vertex 0 is already occupied.

1: Use Algorithm 4 to determine whether vertex $prefix + 2^n - 1$ is occupied.
2: **if** (Vertex $prefix + 2^n - 1$ is occupied) **then**
3:    Return 2^n.
4: **end if**
5: **for** (j := 0; $j < (n-1)$; $j := j + 1$) **do**
6:    Use Algorithm 4 to determine whether vertex $prefix + 2^{n-1} + 2^j$ is occupied.
7:    **if** (Vertex $prefix + 2^{n-1} + 2^j$ is not occupied) **then**
8:       Break the loop.
9:    **end if**
10: **end for**
11: **if** (Vertex $prefix + 2^{n-1} + 2^j$ is not occupied) **then**
12:    Return $2^{n-1} + JOIN(prefix + 2^{n-1}, j)$.
13: **else**
14:    Return $2^{n-1} + JOIN(prefix + 2^{n-1}, n-1)$.
15: **end if**

Ensure: The returned label indicates an empty vertex reserved for the joining peer. Here "return" means that the unoccupied node with least vertex ID is found.

with all neighbors). If a node crashes before it leaves, its neighbors can detect this condition and implement the switch between the crashed node and the last node.

Remark 15 *Like all fully distributed P2P network protocols [19][26], the algorithms presented in this paper assume that the joining node knows at least a node already in the network by some out-of-band means. For example, an online lookup service randomly probes the network and knows polynomial number of network members. The joining node can access the lookup service to know a randomly selected member. The existence of such lookup service poses no anonymity threat to the P2P network (since it has nothing to do with IOI and related anonymity aspects), thus will not be discussed in $\mathcal{PPR}$ anonymity research. As a result, for each member node, it knows its n neighbors (or $n-1$ if the network is not an exact hypercube) plus a node for initiating joining procedure. The extra storage overhead incurred for joining does not affect the polynomial bound.*

All algorithms described in this paper are of at most quadratic time-complexity. First, the routing algorithm shown in Algorithm 3 always ends in $O(n)$ time because the maximal Hamming distance is n. Second, let's assume that the maximal network diameter allowed is $max(n)$ (e.g., 128 in IPv6). Algorithm 1 ends in at most $max(n)$ steps, and each of the step takes at most n steps due to Algorithm 3. Thus Algorithm 1 is of $O(n^2)$ time-complexity. Third, lines 5–10 of Algorithm 2 implement a linear scan over the $(n-1)$ least significant bits of the vertex label, and lines 11–17 of Algorithm 2 implement a recursion over the n-bit label. Both of them end in at most n steps. Since line 6 is of $O(n)$ time-complexity, Algorithm 2 is of $O(n^2)$ time-complexity.

All algorithms described in this paper are of $O(n)$ space-complexity. Each vertex in the hypercube has n neighbors (or in some cases $n-1$ when the total number of vertexes is less than 2^n). In the fully distributed design, each vertex only knows its $O(n)$ neighbors. Certainly, the algorithms can be generalized to acquire connections with c-hop neighbors. The c-hop variant implements a more communication efficient oblivious routing scheme using $O(\frac{1}{c}\cdot n)$ steps by incurring $O(n^c)$ space overhead per node. All such variants feature the same polynomial bound.

5 Conclusions and future work

In this paper we seek to illustrate a newly discovered analogue between anonymity and cryptography. Like message privacy, we show that anonymity can be classified into two major categories: (1) the information-theoretic perfect anonymity and (2) complexity-based anonymity. Like information-theoretic perfect secrecy (which needs 2^n keys to protect 2^n messages against an unbounded cryptanalyst), perfect anonymity is achievable with $O(2^n)$ overhead to resist an unbounded traffic analyst in a network with 2^n nodes, for example, in DC-nets with various network topological settings.

Nevertheless, again like perfect secrecy, this kind of anonymity model is impractical and not useful in the real world. We propose a set of formal notions to model a polynomially bounded adversary and polynomially bounded countermeasures. The newly proposed Probabilistic Polynomial Route ($\mathcal{PPR}$) adversary is an analogue of the conventional Probabilistic Polynomial Time ($\mathcal{PPT}$) adversary, but with network-based abstractions. Then it is possible to define formal notions of anonymity in terms of various capabilities of the attacker. Several advanced attacks, like chosen recipient attack against sender anonymity (CRA), chosen sender attack against recipient anonymity (CSA), and chosen route attack (CRtA), are proposed and studied in this paper. This paper focuses on presenting the basic definitions and how peer-to-peer (P2P) network can be used to realize the new complexity based anonymity model. After this $\mathcal{PPR}$ model is accepted as a valid research foundation, the future work is to design provably secure anonymous protocols on top of the model and to prove the validity of the protocols.

References

[1] R. Berman, A. Fiat, and A. Ta-Shma. Provable Unlinkability against Traffic Analysis. *Financial Cryptography'04, Lecture Notes in Computer Science, Spring-Verlag*, 3110:266–280, 2004.

[2] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. In *Symposium on Foundations of Computer Science (FOCS)*, pages 112–117, 1982.

[3] D. Boneh and M. F. Franklin. Identity Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32:586–615, 2003.

[4] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[5] D. L. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[6] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An Algebraic Approach to Practical and Scalable Overlay Network Monitoring. In *ACM SIGCOMM*, pages 55–66, 2004.

[7] I. Damgård. Collision Free Hash Functions and Public Key Signature Schemes. In D. Chaum and W. L. Price, editors, *EUROCRYPT'87, Lecture Notes in Computer Science 304*, pages 203–216, 1987.

[8] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, 2003.

[9] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), Lecture Notes in Computer Science 2482*, pages 54–68, 2002.

[10] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23th Symposium on the Theory of Computation (STOC)*, pages 542–552, 1991.

[11] S. Dolev and R. Ostrovsky. XOR-trees for Efficient Anonymous Multicast and Reception. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):63–84, 2000.

[12] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[13] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes. In *Public Key Cryptography*, pages 446–465, 2000.

[14] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *21st Symposium on the Theory of Computation (STOC)*, pages 33–43, 1989.

[15] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *22nd Symposium on the Theory of Computation (STOC)*, pages 427–437, 1990.

[16] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In H. Federrath, editor, *DIAU'00, Lecture Notes in Computer Science 2009*, pages 1–9, 2000.

[17] C. Rackoff and D. R. Simon. Noninteractive Zero-knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *CRYPTO'91, Lecture Notes in Computer Science 576*, pages 433–444, 1991.

[18] C. Rackoff and D. R. Simon. Cryptographic defense against traffic analysis. In *Symposium on the Theory of Computation (STOC)*, pages 672–681, 1993.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *ACM SIGCOMM*, pages 161–172, 2001.

[20] S. P. Ratnasamy. A Scalable Content-Addressable Network. PhD Thesis, Computer Science, University of California, Berkeley, Fall 2002.

[21] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[22] A. Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, June 2004.

[23] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), Lecture Notes in Computer Science 2482*, pages 41–53, 2002.

[24] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*, pages 47–53, 1984.

[25] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[26] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM*, pages 149–160, 2001.

[27] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[28] G. S. Vernam. Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications. *Journal American Institute of Electrical Engineers*, XLV:109–115, 1926.

[29] L. von Ahn, A. Bortz, and N. J. Hopper. k-Anonymous Message Transmission. In V. Atluri and P. Liu, editors, *10th ACM Conference on Computer and Communications Security (CCS 2003)*, pages 122–130, 2003.

[30] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

A Notations

In this paper $x \in_R X$ denotes that x is uniformly selected from the set X. For bitwise operations, $\oplus$ denotes bitwise exclusive-OR, $\&$ denotes bitwise AND, $||$ denotes string concatenation, and $|x|$ denotes bit-length for bit-string x or cardinality for set x.

The term "encipher" (or "decipher") refers to protecting plaintexts using one-time pads. The term "encryption" (or "decryption") refers to protecting plaintexts using trapdoor one-way permutations.

B Hypercube

Definition 16 ***(Hypercube)*** *Let Q_n be a hypercube of n-dimension. Q_n is inductively constructed by the following algorithm:*

- *Q_0 is a single point with 2^0 vertex.*
- *Q_1 is a line segment with 2^1 vertexes.*
- *Given Q_{n-1} with 2^{n-1} vertexes, we label each vertex of Q_{n-1} with $(n-1)$-bit binary string. Then Q_{n-1} is duplicated as Q'_{n-1}. In the original Q_{n-1}, each vertex label is prefixed with 0. In Q'_{n-1}, each vertex label is prefixed with 1. Q_n is constructed from Q_{n-1} and Q'_{n-1} by connecting them: A vertex in Q_{n-1} is connected with a vertex in Q'_{n-1} if their labels differ only at the most significant bit.* ■

Q_n is comprised of two Q_{n-1}'s: the most significant bit (MSB) of one Q_{n-1} is 0, and the MSB of the other Q_{n-1} is 1. We will called the former one "lowerhalf" and the latter one "upperhalf". Some vertexes in the upperhalf are allowed to be empty in our hypercube construction.

Algorithm 3 Oblivious routing in hypercube

Require: No global view of the network is allowed. Each node only knows its neighbors. The algorithm is initiated on any occupied vertex V'.

```
currentV := V'.
loop
    nextV := flip any different bit between currentV and V.
    if (nextV == V) then
        Break the loop. Find out whether my neighbor V is occupied.
    end if
    if (nextV is unoccupied) then
        Break the loop. V is not reachable thus unoccupied.
    end if
    currentV := nextV. Go from currentV to nextV.
end loop
```

Ensure: Either declare V is not reachable and unoccupied, or reach V. The algorithm ends in at most d steps, where d is the Hamming distance between V' and V.

If the network is an exact hypercube, we can flip *any* different bit in Algorithm 3. However, in practice the upperhalf of Q_n may have empty vertexes and the lowerhalf is fully occupied, thus the links crossing the upperhalf and the lowerhalf must be taken at the very beginning and the very end, if there is any. This means that (1) the most significant bit should always be flipped at the first step if V' is in upperhalf (i.e., MSB of V' is 1, this directs the entire routing into the lowerhalf), and (2) the MSB is flipped again at the last step if V is in upperhalf.

The following algorithm is provided for Algorithm 2.

Algorithm 4 Oblivious routing in hypercube Q_n with $prefix$

Require: Same as Algorithm 3, but all bits to the left of the least-significant one in the $prefix$ (including the one bit) will be ignored.

  $currentV$:= V'.

  loop

    $nextV$:= flip any different bit between $currentV$ and V except all the bits to the left of the least-significant 1 in the $prefix$.

    if ($nextV == V$) **then**

      Break the loop. Find out whether my neighbor V is occupied.

    end if

    if ($nextV$ is unoccupied) **then**

      Break the loop. V is not reachable thus unoccupied.

    end if

    $currentV$:= $nextV$. Go from $currentV$ to $nextV$.

  end loop

Ensure: Same as Algorithm 3.
