k-Anonymous Message Transmission

Luis von Ahn Andrew Bortz Nicholas J. Hopper
Computer Science Department
Carnegie Mellon University
{biglou,abortz,hopper}@cs.cmu.edu

September 3, 2003

Abstract

Informally, a communication protocol is *sender k - anonymous* if it can guarantee that an adversary, trying to determine the sender of a particular message, can only narrow down its search to a set of k suspects. *Receiver k-anonymity* places a similar guarantee on the receiver: an adversary, at best, can only narrow down the possible receivers to a set of size k. In this paper we introduce the notions of sender and receiver k-anonymity and consider their applications. We show that there exist *simple* and *efficient* protocols which are k-anonymous for both the sender and the receiver in a model where a polynomial time adversary can see all traffic in the network and can control up to a constant fraction of the participants. Our protocol is provably secure, practical, and does not require the existence of trusted third parties. This paper also provides a *conceptually simple* augmentation to Chaum's DC-Nets that adds robustness against adversaries who attempt to disrupt the protocol through perpetual transmission or selective non-participation.

1 Introduction

Anonymous or untraceable communication protocols have been studied extensively in the scientific literature (e.g. [4, 5, 16, 19]). These protocols address the problem of concealing who communicates with whom, as in the case of letters from a secret admirer. The adversary, trying to determine the sender or recipient of a message, is allowed to see all the communications in the network, so a protocol for anonymous communication even allows Bob to send a secret love letter to Eve, the network administrator. If used in practice, anonymous communication would have many important applications such as guaranteeing anonymous crime tip hotlines or allowing "whistle blowers" inside corrupt organizations to leak secrets to the press.

The goal is usually to guarantee full anonymity: an adversary looking at the communication patterns should not learn *anything* about the origin or destination of a particular message. To gain efficiency we concentrate on a weaker goal, k-anonymity: the adversary is able to learn something about the origin or destination of a particular message, but cannot narrow down its search to a set of less than k participants. In other words, k-anonymity guarantees that in a network with n honest participants, the adversary is not able to guess the sender or recipient of a particular message with probability non-negligibly greater than $1/k$, where k is a constant smaller than, but otherwise not related to n. We show that, in our adversarial model, there exists a k-anonymous communication protocol that is far simpler and more efficient than any known fully anonymous communication protocol.

While k-anonymity is a weaker guarantee, it is still sufficient for a variety of applications. For example, in the United States legal system, 2-anonymity would be enough to cast "reasonable doubt," thus invalidating a criminal charge, while 3-anonymity would be enough to invalidate a civil charge, in the absence of other evidence. This is especially relevant after a federal judge in the United States ordered Verizon Communications, a large ISP, to disclose the identity of an alleged peer-to-peer music pirate — a legal decision that could make it easier for the music industry to crack down on file swapping [3]. If the participants in the peer-to-peer network were communicating k-anonymously, the music industry could not prosecute individuals in this manner. k-anonymity is also enough for the protection of privacy in everyday transactions, as it effectively breaks data profiling techniques.[1]

The protocol presented in this paper is extremely efficient and provably secure in a strong adversarial model: we assume that the adversary can see all the communications between the participants and can control any constant fraction (up to 1/2) of the participants. Participants owned by the adversary can act arbitrarily and attempt to ruin the communication protocol in any possible way — i.e., the adversary not only tries to determine the sender or recipient of particular messages, but also tries to render the anonymous communication protocol useless. We assume the adversary is *computationally bounded* (polynomial time) and *non-adaptive* (the adversary must choose which participants to corrupt before the execution of the protocol). We also assume that the network is not adversarially unreliable: messages between communicating parties are always delivered.[2] This assumption is mostly for simplicity, as our protocol can be used on top of schemes that guarantee reliable communication in an adversarially unreliable setting (e.g. [7]) at the expense of efficiency.

For the most part, the study of anonymous communication has focused on efficiency rather than on provable security, and many of the current systems fail when confronted by sufficiently powerful adversaries [20]. Our protocol is provably secure in a strong adversarial model and achieves part of its efficiency by allowing the anonymity guarantee to vary: k can be any number between 1 and n (the number of participants in the network). The efficiency of our protocol is related to the size of k and for small values of k the protocol is efficient enough to be used in practice. However, it is important to mention that, while k-anonymity is sufficient in many settings, there are cases where full anonymity is required (e.g. ransom notes). If k equals n (i.e., in the case of full anonymity) our protocol is simpler and as efficient as any known protocol that is provably secure in our adversarial model. In this way, our protocol can also be viewed as a conceptually simple and efficient augmentation to Chaum's DC-Nets that adds robustness against adversaries who attempt to disrupt the protocol through perpetual transmission or selective non-participation.

Related Work

Below we describe a few of the most influential solutions to the anonymous communication problem and compare them to our proposal.[3]

DC-Nets [4, 19]. DC-Nets is an anonymous broadcast protocol that bases its anonymity on the strength of a secure multiparty sum computation. In this fashion, it is one of the few systems that provides provable security in the absence of trusted parties. Although the original system by

[1]The concept of k-anonymity in fact comes from the privacy literature [18].

[2]As long as the network is not adversarially unreliable, there exist protocols (such as TCP) that provide reliable delivery.

[3]This section is only meant to provide a sample of the previous work so as to put our proposal in context; it is not meant to provide a complete description of the literature. See [9] for a more thorough listing.

Chaum [4] was susceptible to certain attacks, a later variant by Waidner [19] provides an elaborate system of traps and commitments that guarantees robustness and anonymity. However, the poor scalability of DC-Nets makes it unsuitable for medium or large-scale use. In particular, in a network of n users, DC-Nets incurs a cost of $\Omega(n^3)$ protocol messages per anonymous message in every case. Our protocol is similar to DC-Nets, but with a much simpler and efficient method of guaranteeing robustness, better scaling properties, and the ability to amortize message complexity over several anonymous messages. Our adversarial model is similar to that assumed in the DC-Nets literature except that we restrict the adversary to run in polynomial time.

Mix-Nets [5] and Onion Routing. Mix-Nets, introduced by David Chaum in 1981, was one of the first concepts for anonymizing communication. The idea is that a trusted "Mix" shuffles messages and routes them, thus confusing traffic analysis. Chaining Mixes together to form a path, combined with Mix-to-Mix (Onion Routing) and end-to-end encryption, offers a form of provable security against a completely passive adversary [5]. Mix-Nets requires the existence of semi-trusted nodes: security is guaranteed as long as one Mix (out of a small constant number of them) is honest.

In every Mix-Nets proposal, an active adversary who participates in the system is able to degrade the anonymity of selected messages and users with non-negligible probability [14], and also degrade efficiency through excessive, anonymous usage of its capabilities [12] and selective, undetectable non-participation [20].

Compared to Mix-Nets protocols, our solution incurs fewer network latencies, requires no special trusted nodes, and is provably secure against non-participating active adversaries. However, our solution incurs higher communication and computational complexity.

Crowds [16]. Similar to Mix-Nets, Crowds provides paths to disguise the originator of a message. Unlike Mix-Nets, however, paths in Crowds are determined randomly by the machines through which a message passes, rather than by the originator of the message. Crowds provides sender *probable innocence* against an adversary who controls a certain fraction of the participants.[4] However, Crowds provides no protection against a global eavesdropper. k-anonymity can be seen as a further refinement of probable innocence and in particular, our protocol for the case of 2-anonymity is competitive with Crowds in terms of round complexity, slightly worse in communication complexity and incurs much heavier computational costs, while providing provable security in a much stronger adversarial model.

CliqueNet [17]. CliqueNet combines small DC-Nets with a routing layer to mitigate the scalability problems of DC-Nets while also preserving some of its anonymity guarantees. CliqueNet has the undesirable feature, however, that an adversary who controls ℓ network nodes can completely compromise the anonymity of $\ell - 1$ other nodes of its choice. Furthermore, CliqueNet's routing layer induces a high amount of unnecessary network latency and is not secure against non-participation, allowing an adversary who controls a few nodes to partition the network. Our protocol is similar to CliqueNet in that we also divide the network into small DC-Nets-like components, but different in that we provide provable security against strong adversaries.

[4] A protocol provides sender probable innocence if the receiver cannot identify the sender with probability greater than 1/2.

Organization of the Paper

Section 2 presents the basic cryptographic notions and definitions we will need for the paper. Section 3 introduces the definitions for k-anonymous communication, Section 4 introduces the novel protocol that achieves k-anonymity for both the sender and the receiver, and Section 5 delineates how to construct a communications network that can guarantee k-anonymity. Finally, Section 6 concludes with a discussion and some open questions.

2 Preliminaries

2.1 Notation

A function $\mu : \mathbb{N} \to [0,1]$ is said to be *negligible* if for every $c > 0$, for all sufficiently large n, $\mu(n) < 1/n^c$. Let S be a set, then $x \leftarrow S$ denotes the action of choosing x uniformly from S. U_k denotes the set of k-bit strings. We denote the set of integers $\{1, \ldots, n\}$ by $[n]$. We will use $\mathbb{Z}_m$ to denote the additive group of integers modulo m, and $\mathbb{Z}_m^*$ to denote the multiplicative group of integers modulo m. When we say *split* $x \in \mathbb{Z}_m$ *into* n *random shares* $s_1, \ldots, s_n$ we mean choose $s_1, ...s_{n-1}$ uniformly at random from $\mathbb{Z}_m$ and set $s_n = x - (s_1 + \cdots + s_{n-1}) \bmod m$. For parties P and Q, the notation $P \longrightarrow Q : M$ denotes party P sending message M to party Q.

2.2 The Model

We assume a network of n parties $\{P_1, \ldots, P_n\}$, of which a fraction $\beta < 1/2$ are controlled by a non-adaptive polynomial time adversary, who may also monitor the communications between all parties. We assume the existence of a trusted public-key infrastructure which allows secure authenticated channels between all pairs of parties. Otherwise, parties under the control of the adversary may behave arbitrarily (the remaining *honest* parties are constrained by the protocol). We also assume that the network is reliable: messages between parties are always delivered.

2.3 Pedersen Commitments

Let p and q be primes such that q divides $p - 1$, and let $g, h \in \mathbb{Z}_p^*$ have order q. (It is easy to see that both g and h generate the unique subgroup of order q in $\mathbb{Z}_p^*$.) The following commitment scheme will be used throughout the paper; it is due to Pedersen [13] and is based on the difficulty of finding $\log_g(h)$ (all the multiplications are over $\mathbb{Z}_p^*$):

- To commit to $s \in \mathbb{Z}_q$, choose r uniformly from $\mathbb{Z}_q$ and output $C_r(s) = g^s h^r$.
- To open the commitment, simply reveal s and r.

For any s, the commitment $C_r(s) = g^s h^r$ is uniformly distributed over the unique subgroup of order q in $\mathbb{Z}_p^*$, so that $C_r(s)$ reveals no information about s. Furthermore, the committer cannot open a commitment to s as $s' \neq s$ unless she can find $\log_g(h)$. Hence, this is a perfectly hiding, computationally binding commitment scheme. In addition, this commitment scheme is *homomorphic:* given commitments $C_{r_1}(s_1)$ and $C_{r_2}(s_2)$, we have that $C_{r_1}(s_1)C_{r_2}(s_2) = C_{r_1+r_2}(s_1 + s_2)$.

2.4 Zero-Knowledge Proofs

Informally, a zero-knowledge proof is a protocol which allows a *prover* program P to convince a *verifier* program V of the veracity of a statement while giving the verifier no additional knowledge. For now we will only require security in the case of an *honest verifier* (i.e., the verifier follows the program V). There exist standard techniques ([10], [2]) to convert the particular type of honest-verifier zero-knowledge proof that we will use into a proof which is secure even against a dishonest verifier.

Definition 1. *A protocol* (P, V) *is* honest verifier zero-knowledge *if there is an efficient program* S *(a* simulator*) such that the output of* $S(x)$ *and the view of* V *upon interaction with* $P(x)$ *are indistinguishable.*

An example is the following protocol for proving *knowledge* of the discrete logarithm of $x = h^r \bmod p$ (where q divides $p-1$ and p, q are prime) originally due to Chaum *et al.* [6]:

Protocol 1. *Zero-knowledge proof of knowledge for discrete logarithms*

1. P picks $\sigma \leftarrow \mathbb{Z}_q$,
   $P \longrightarrow V \ : \ y = h^\sigma \bmod p$
2. $V \longrightarrow P \ : \ z \leftarrow \mathbb{Z}_q$
3. $P \longrightarrow V \ : \ w = rz + \sigma \bmod q$
4. V accepts if $x^z y = h^w$

The honest-verifier simulator for this protocol first selects the values $z, w \leftarrow \mathbb{Z}_q$ and sets $y = h^w / x^z \bmod p$, then outputs the conversation y, z, w. A prover can cheat in this protocol only with very small probability, $1/q$.

2.5 Secure Multiparty Sum

A secure multiparty addition protocol allows parties P_1,
$\ldots, P_n$, each with a private input $X_i \in \mathbb{Z}_m$, to compute $X = X_1 + \ldots + X_n$ in such a way that P_i, regardless of its behavior, learns nothing about X_j, $i \neq j$, except what can be derived from X. The following commonly-known scheme implements secure multiparty addition: each party P_i splits X_i into n random shares $s_{i,1}, \ldots, s_{i,n}$ such that $\sum_j s_{i,j} = X_i$ and sends share $s_{i,j}$ to party j; later all parties add every share that they have received and broadcast the result. It is easy to see that the sum of all broadcasts equals $X_1 + \ldots + X_n$, and that it is impossible for party j to learn anything about X_i (for $i \neq j$).

For the rest of this paper, we use the following modification of the above scheme. The commitments help ensure that all parties adhere to the protocol (e.g., parties shouldn't be able to cheat by sending inconsistent shares):

Protocol 2. *Secure Multiparty Sum*

1. **Commitment Phase:**
   - P_i splits $X_i \in \mathbb{Z}_q$ into n random shares $s_{i,1}, ..., s_{i,n}$
   - P_i chooses $r_{i,j} \leftarrow \mathbb{Z}_q$

- P_i computes commitments $C_{i,j} = C_{r_{i,j}}(s_{i,j})$
- P_i broadcasts $\{C_{i,j} : 1 \leq j \leq n\}$

2. **Sharing Phase:**
   - For each $j \neq i$,
     $P_i \longrightarrow P_j \ : \ (r_{i,j}, s_{i,j})$.
   - P_j checks that $C_{r_{i,j}}(s_{i,j}) = C_{i,j}$

3. **Broadcast Phase:**
   - P_i computes the values $R_i = \sum_j r_{j,i} \bmod q$ and $S_i = \sum_j s_{j,i} \bmod q$
   - P_i broadcasts (R_i, S_i)
   - All players check that $C_{R_i}(S_i) = \prod_j C_{j,i} \bmod p$

4. **Result:**
   Each player computes the result as $X = \sum_i S_i \bmod q$, computes $R = \sum_i R_i \bmod q$ and checks that $C_R(X) = \prod_{i,j} C_{i,j} \bmod p$

Note that as long as every party transmits something, the broadcast does not need to be reliable (i.e. it does not matter if an adversary conspires to make two different players get different values); because of the use of commitments, either some value fails a check for some honest player, or the final result is identical to another instance of the protocol where the adversary does not send different messages (a rigorous proof of this fact appears in the Appendix). This protocol is susceptible to only one kind of disruptive attack: *selective non-participation*, in which an adversary either does not send some protocol messages to a participant or claims that it has not received any message from that participant. As the protocol is stated, there is no way to tell whether the sender failed to send a message or the receiver is falsely claiming that it didn't receive it. Selective non-participation will be dealt with in later sections.

Secure multiparty addition and anonymous communication are related (an observation which seems to be due to David Chaum and forms the basis of DC-Nets), in that a protocol for secure multiparty addition can be used to perform anonymous broadcast. Assume that party j wants to broadcast the message $X_j \neq 0$ anonymously, while the other parties do not wish to broadcast anything; then by performing a multiparty addition with $X_i = 0$ (for $i \neq j$), all the parties learn $X_1 + \ldots + X_n = X_j$, but nobody learns where X_j came from. If more than one party tries to transmit at the same time, however, a collision occurs and the parties have to try again. For this reason DC-Nets use a complicated reservation mechanism to keep the adversary from jamming the channel: jamming can occur when the adversary controls a participant and simply sends a message at every time step. Our protocol is also based on secure multiparty sum computations, but one of the novel aspects of our work is the relatively simple mechanism that we use to prevent the adversary from jamming the channel.

3 Definitions

An anonymous communication protocol for message space $\mathcal{M}$ is a computation among n parties $P_1, \ldots, P_n$, where each P_i starts with a private input $(msg_i, p_i) \in (\mathcal{M} \times [n]) \cup \{(\mathtt{nil}, \mathtt{nil})\}$, and each party terminates with a private output from $\mathcal{M}^*$. To communicate, time will be split into *rounds* and the protocol will be run at each round. Intuitively, at the end of a round each P_i should learn the set of messages addressed to him ($\{msg_j : p_j = i\}$), but not the identity of the senders.

We let $H \subset \{P_1, \ldots, P_n\}$ denote the set of honest parties. We denote by $\mathcal{P}(P_1(msg_1, p_1), \ldots, P_n(msg_n, p_n))$ the random variable distributed according to the adversary's *view* of the protocol $\mathcal{P}$ when each P_i has input (msg_i, p_i). We denote by $\mathcal{P}(P_i(msg_i, p_i), *)$ the adversary's view of $\mathcal{P}$ when P_i has input (msg_i, p_i) and the other inputs are set arbitrarily.

3.1 Full Anonymity

Definition 2. *A protocol $\mathcal{P}$ is* sender anonymous *if for every pair $P_i, P_j \in H$, and every pair $(msg, p) \in (\mathcal{M} \times [n]) \cup \{(\texttt{nil}, \texttt{nil})\}$, $\mathcal{P}(P_i(msg, p), *)$ and $\mathcal{P}(P_j(msg, p), *)$ are computationally indistinguishable.*

That is, a protocol is sender anonymous if the adversary may not distinguish between any of the honest parties as the sender of a message, regardless of who the receiver is; i.e., the adversary "gains no information" about the sender.

Definition 3. *A protocol $\mathcal{P}$ is* receiver anonymous *if for every $P' \in H$, for every $msg \in \mathcal{M}$ and every $P_i, P_j \in H$, $\mathcal{P}(P'(msg, P_i), *)$ and $\mathcal{P}(P'(msg, P_j), *)$ are computationally indistinguishable.*

According to the previous definitions, the trivial protocol in which no party transmits anything is both sender and receiver anonymous. Non-triviality is captured by Definition 6 below.

Assuming that the protocol is non-trivial (i.e., useful), sender anonymity requires every honest party, even if they have no message as an input, to send at least one *protocol* message per anonymous message delivered.Thus any protocol which is sender anonymous has a worst-case lower bound of n protocol messages per input message, since *in the worst case*, all parties but one have input $(\texttt{nil}, \texttt{nil})$. If n is large, this lower bound makes it unlikely that a system providing full anonymity can be fielded in practice.

3.2 k-Anonymity

Definition 4. *A protocol $\mathcal{P}$ is* sender k-anonymous *if it induces a partition $\{V_1, \ldots, V_l\}$ of H such that:*

1. *$|V_s| \geq k$ for all $1 \leq s \leq l$; and*
2. *For every $1 \leq s \leq l$, for all $P_i, P_j \in V_s$, for every $(msg, p) \in (\mathcal{M} \times [n]) \cup \{(\texttt{nil}, \texttt{nil})\}$, $\mathcal{P}(P_i(msg, p), *)$ and $\mathcal{P}(P_j(msg, p), *)$ are computationally indistinguishable.*

That is, each honest party's messages are indistinguishable from those sent by at least $k - 1$ other honest parties.

Definition 5. *A protocol $\mathcal{P}$ is* receiver k-anonymous *if it induces a partition $\{V_1, \ldots, V_l\}$ of H such that:*

1. *$|V_s| \geq k$ for all $1 \leq s \leq l$; and*
2. *For every $1 \leq s \leq l$, for all $P_i, P_j \in V_s$, for every $P' \in H$, $msg \in \mathcal{M}$: $\mathcal{P}(P'(msg, P_i), *)$ and $\mathcal{P}(P'(msg, P_j), *)$ are computationally indistinguishable.*

That is, each message sent to an honest party has at least k indistinguishable recipients.

3.3 Robustness

In addition to the anonymity guarantees, we will require that the communication protocol be robust against an adversary trying to render it useless. We capture this intuition with the notion of *robustness.*

Definition 6. *Let $\alpha \in [0,1]$. A protocol $\mathcal{P}$ is* α-robust *if in each round, the protocol satisfies at least one of the following conditions:*

Fairness: *For all $P' \in H$ and for all $(msg, i) \in (\mathcal{M} \times [n])$, if P' has as input (msg, i), the probability (over the randomness of P') that party P_i receives msg is at least α*

Detection: *The set S of parties who deviate from $\mathcal{P}$ is non-empty and there is a single $P_i \in S$ such that for all $P_j \in H$, P_j outputs P_i.*

That is, for every round, either the protocol was fair, or an adversarially controlled party was exposed.

4 The Protocol

Our solution to the k-anonymous message transmission problem is similar to Chaum's [4] DC-Nets but features two important innovations.

First, we partition the n parties into smaller groups of size $M = O(k)$ such that with high probability k members of each group are honest. Each group performs essentially the multiparty sum protocol described in Section 2, where the input X_i is of the form (msg, g), a pair describing the message msg to be transmitted and the group g of the receiver. This guarantees receiver k-anonymity as well as sender k-anonymity, because sending to one member of group g is identical to sending to any other member of g, and there are always k honest participants in each group.

Second, each group runs $2M$ copies of the multiparty sum protocol in parallel, restricting each party to transmit in at most one parallel copy, so as to provide fairness. We give a protocol which allows the detection of at least one non-conforming party in each round where access to this shared channel was not fair. Since each group has only $O(k)$ non-conforming parties, an adversary can only cause $O(k)$ protocol failures in each group, and no protocol failure compromises the anonymity of any honest party. In comparison, previous solutions built around DC-Nets may involve letting a protocol failure go undetected or compromising the anonymity of a message.

4.1 Description

The protocol will be described in steps for ease of exposition. The first, Protocol 3, will not be secure against non-participation.

4.1.1 Transmission

Protocol 3. *k-AMT*

Precondition: Assume that the n parties are partitioned into groups of size M, with each group having at least k honest participants (in Section 5 we discuss how this precondition is met). Below are the instructions to be performed by each group individually. For notational simplicity, we denote the parties in the current group by $P_1, \ldots, P_M$, and the public encryption keys of these parties by $PK_1, \ldots, PK_M$. "Broadcast" means to send to every other member of the group.

Input: Each party P_i in the group has input g_i, the group the receiver belongs to, and msg_i, a message. (msg_i, g_i) will be interpreted as an element of $\mathbb{Z}_q$, where q is a large prime that divides $p-1$ (p is also a prime). We identify $(msg_i, g_i) = (\mathtt{nil}, \mathtt{nil})$, indicating "no message this round," with $0 \in \mathbb{Z}_q$.

1. **Commitment Phase:**
   - P_i chooses $l \leftarrow [2M]$ and sets $X_i[l] = (msg_i, g_i)$ and $X_i[t] = 0$ for $t \neq l \in [2M]$
   - P_i splits $X_i[t] \in \mathbb{Z}_q$ into M random shares $s_{i,1}[t], \ldots, s_{i,M}[t]$ for $t \in [2M]$
   - P_i chooses $r_{i,j}[t] \leftarrow \mathbb{Z}_q$ for all $j \in [M], t \in [2M]$
   - P_i computes commitments $C_{i,j}[t] = C_{r_{i,j}[t]}(s_{i,j}[t])$
   - P_i broadcasts $\{C_{i,j}[t] : j \in [M], t \in [2M]\}$
2. **Sharing Phase:**
   - For each $j \neq i$,
     $P_i \longrightarrow P_j \ : \ \{(r_{i,j}[t], s_{i,j}[t]) : t \in [2M]\}$
   - P_j checks that $C_{r_{i,j}[t]}(s_{i,j}[t]) = C_{i,j}[t]$
3. **Broadcast (only within the group) Phase:**
   - P_i computes the values $R_i[t] = \sum_j r_{j,i}[t] \bmod q$ and $S_i[t] = \sum_j s_{j,i}[t] \bmod q$
   - P_i broadcasts $\{(R_i[t], S_i[t]) : t \in [2M]\}$
   - All players check that $C_{R_i[t]}(S_i[t]) = \prod_j C_{j,i}[t]$
     $\bmod p$
4. **Result:**
   Each player computes the result as $X[t] = \sum_i S_i[t] \bmod q$, computes $R[t] = \sum_i R_i[t] \bmod q$ and checks that $C_{R[t]}(X[t]) = \prod_{i,j} C_{i,j}[t] \bmod p$
5. **Transmission Phase:**
   For each $X[t] \neq 0$, each P_i interprets $X[t]$ as a pair $(Msg[t], G[t])$ and sends $Msg[t]$ to every member of $G[t]$

4.1.2 Fairness

Suppose at the conclusion of the transmission phase, at most M of the $2M$ values $X[t]$ were non-zero. Then this execution was *fair*: each P_i had probability at least $1/2$, over its own choices, of successfully transmitting msg_i. On the other hand, if *more* than M of the $X[t]$ were non-zero, then at least one P_i had more than one $X_i[t] \neq 0$. We now describe an *honest verifier* statistical zero-knowledge proof that allows each honest party to prove that they set at most one $X_i[t]$ to a non-zero value, assuming it is hard to compute $\log_g(h)$ (this allows the honest players to identify at least one party P_i with more than one $X_i[t]$ not equal to zero).

Informally, this protocol uses the well-known "cut-and-choose" technique: player P_i prepares new commitments $C'_i[t]$ to the values $X_i[t]$ and randomly permutes them. Then the verifier may choose either to have P_i open $2M-1$ of the (permuted) $C'_i[t]$ values to zero, or to have P_i reveal the permutation and prove (in zero-knowledge) that he can open the commitments $C'_i[t]$ and $C_i[t]$ (for each $t \in [2M]$) to the same value.

Protocol 4. *Zero-knowledge proof of fairness*

1. P_i chooses $r'[t] \leftarrow \mathbb{Z}_q, t \in [2M]$, and $\pi \leftarrow \mathbb{S}_{2M}$ (a permutation on $\{1, \ldots, 2M\}$). Define

$$\begin{aligned} \rho_i[t] &= \sum_j r_{i,j}[t] \ , \\ C_i[t] &= \prod_j C_{i,j}[t] \bmod p = C_{\rho_i[t]}(X_i[t]) \ , \\ \rho'_i[t] &= r_i[t] + r'[t] \bmod q \ , \\ C'_i[t] &= C_i[t] h^{r'[t]} \bmod p = C_{\rho'_i[t]}(X_i[t]) \end{aligned}$$

   $P_i \longrightarrow V \ : \ \langle \kappa[t] = C'_i[\pi(t)] \rangle_{t=1,\ldots,2M}$

2. $V \longrightarrow P_i : b \leftarrow \{0, 1\}$

3. If $b = 0$, then:

   - P_i sets l such that $X_i[l] \neq 0$ if l exists, or chooses $l \leftarrow \{1, \ldots, 2M\}$ otherwise.
     $P_i \longrightarrow V : \langle \xi[t] = \rho'_i[\pi(t)] \rangle_{\pi(t) \neq l}$
   - V accepts iff $h^{\xi[t]} = \kappa[t] \bmod p$ for all $t \neq \pi^{-1}(l)$

   Otherwise, P_i proves that C' is a commitment to a permutation of C by revealing π and proving knowledge of the discrete log of $x[t] = C'_i[t]/C_i[t] = \kappa[\pi^{-1}(t)]/C_i[t]$:

   - P_i picks values $\sigma[t] \leftarrow \mathbb{Z}_q$,
     $P_i \longrightarrow V \ : \ \pi, \langle y[t] = h^{\sigma[t]} \bmod p \rangle_t$
   - $V \longrightarrow P_i \ : \ \langle z[t] \leftarrow \mathbb{Z}_q \rangle_t$
   - $P_i \longrightarrow V \ : \ \langle w[t] = r'[t] z[t] + \sigma[t] \bmod q \rangle_t$
   - V accepts if $x[t]^{z[t]} y[t] = h^{w[t]}$, for all $t \in [2M]$

The above protocol is public-coin, honest-verifier statistical zero knowledge. In practice, we may implement the verifier by calls to a cryptographic hash function and obtain security in the Random Oracle Model [2], or the verifier may be implemented by the remaining parties through a subprotocol in which each party non-malleably commits to random bits and then reveals the bits; the randomness used is then the exclusive-or of each party's random string. So long as there is one honest verifier this approach will work: a party which refuses to participate in this subprotocol can be recognized as the cheating party.

4.1.3 Non-participation

Unfortunately, the previous protocol neglects the ability of an adversary to refuse to transmit data altogether. In fact, this has typically been the hardest of all scenarios to cope with. In such a situation, it is impossible to arbitrate correctly as to whether the required sender did not send a message, or the alleged receiver is lying about not receiving the message. An augmentation is required to Protocol 3 in order to deal with this situation:

Protocol 5. *k-AMT2*

2. **New Sharing Phase:**

- For each $j \neq i$,
  $P_i \longrightarrow P_j \ : \ \{E_{PK_l}(r_{i,l}[t], s_{i,l}[t]) : l \in [M], t \in [2M]\}$.
- P_j checks that $C_{r_{i,j}[t]}(s_{i,j}[t]) = C_{i,j}[t]$

After each phase of Protocol 3:

1. **Timeout Step:** For all P_j failing to receive a required message from P_i after the timeout period, P_j sends a signed "timeout" message $T\{i,j\}$ to every group member.

2. **Correction Step:** For each $i' \neq j$, $l, j' \in [M]$, $t \in [2M]$:

   - if the Commitment phase has begun,
     $P_{i'} \longrightarrow P_j \ : \ \{C_{l,j'}[t]\}$
   - if the Sharing phase has begun,
     $P_{i'} \longrightarrow P_j \ : \ \{E_{PK_j}(r_{l,j}[t], s_{l,j}[t])\}$
   - if the Broadcast phase has begun,
     $P_{i'} \longrightarrow P_j \ : \ \{(R_l[t], S_l[t])\}$
   - Finally,
     $P_{i'} \longrightarrow P_j \ : \ \{T\{a,b\} : (P_a \rightarrow P_i : T\{a,b\}) \ \}$

Here, $E_K(m)$ denotes the public-key encryption of m with public key K, where E is a semantically-secure public key encryption scheme. Under this augmentation, the message and round complexity of the protocol increase by a factor of at most 2, and the bit complexity increases by a factor of M. For space considerations, we omit the full description and analysis of two alternative schemes which avoid this factor of M increase in bit complexity. The first reduces bit complexity by modifying the Correction Step to the Commitment Phase (the first bullet of step 2 above). Rather than having each honest participant send all M commitment matrices to every other participant, each honest participant sends only a randomly chosen subset of size $\log_e M$. The robustness of the protocol is then decreased by an additive factor of $1/M$. The second scheme works by tracking which pairs of participants are unwilling to communicate and constructing broadcast trees which avoid these links at the expense of extra rounds; the key observation is that, when some participant is no longer connected to some complete subgraph of size k he can be dropped from the network, so that an adversary cannot arbitrarily increase the round complexity.

4.2 Analysis

4.2.1 Robustness

Let us now consider the success of all possible attacks against the robustness of the protocol. Note that whenever an investigation is warranted (any check fails), a simple subprotocol is executed wherein every player reliably broadcasts every received *broadcast* from the other players. If a party is found to have sent different signed broadcasts, it is identified as the cheater. If not, the investigation continues.

The simplest possible deviation is for an adversary to attempt to jam the channel by transmitting in more than one slot. However, if access to the channel was not fair, then this is detected with high probability. Since we have already verified that all broadcasts were made correctly, then each party has the same commitment matrix (the first broadcast) for every other player. Therefore, the zero-knowledge subprotocol will detect the cheater with negligible chance of failure.

Theorem 1. *Protocol 4 is sound: if for some i, there exist $t \neq t'$ such that $X_i[t] \neq 0$ and $X_i[t'] \neq 0$ then $|\Pr[V \text{ accepts}] - \frac{1}{2}|$ is negligible.*

Proof. (Sketch) Suppose V chooses $b = 0$; then, if the commitments C'_i are formed correctly P_i must compute $\log_g h \pmod p$ in order to open one of $C'_i[t], C'_i[t']$ to zero. If computing discrete logarithms modulo p is hard, then this happens with negligible probability. Likewise, if V chooses $b = 1$, then if the commitments C'_i are malformed, P_i must compute $\log_h g$ in order to make V accept (by the soundness of the discrete logarithm subprotocol in step 3). So for the honest V, regardless of the formation of the commitments C'_i, P_i has probability at most $1/2$ plus a negligible factor of convincing V to accept. □

Theorem 2. *Protocol 4 is honest-verifier zero-knowledge.*

Proof. (Sketch) We exhibit a simulator for the honest-verifier case: flip a coin representing the bit b in step 2. If $b = 0$, form the commitments $C'_i[t] = C_{r'[t]}(0)$ from step 1, choose a random $l \in 1, \ldots, 2M$ and reveal $r'[1], \ldots, r'[l-1], r'[l+1], \ldots, r'[2M]$ in step 3. If $b = 1$, form the commitments $C'_i[t]$ in the same manner as the honest prover, and use the honest-verifier simulator for the discrete logarithm protocol in step 3. □

Given that incorrect broadcasts will always be detected, and non-participation is dealt with, the only other possible deviation is to send incorrect data. However, because of the use of commitments, every piece of data is either a commitment that will have to be opened, or the opening of an already transmitted commitment (or commitment product). Therefore, this deviation will be detected as long as breaking the commitment scheme is hard.

4.2.2 Anonymity

Theorem 3. *If group G has at least k honest parties, then Protocol 3 is sender k-anonymous for group G.*

Proof. (Sketch) In each parallel round, the multiparty sum protocol guarantees that no adversary may determine the inputs of any honest parties; thus the adversary may not distinguish between the case that $X_i[t] = 0$ and $X_i[t] = Msg[t]$ for any honest party. □

Theorem 4. *If every group G has at least k honest parties, then Protocol 3 is receiver k-anonymous.*

Proof. (Sketch) Each message sent to an honest party P_i is received by all parties in P_i's group; since there are at least k honest parties in this group, the adversary cannot distinguish between these parties as the recipients. □

Theorem 5. *If the precondition for Protocol 3 holds, Protocol 4 and Protocol 5 together give a $\frac{1}{2}$-robust k-anonymous transmission protocol.*

4.2.3 Efficiency

Because we detect cheaters with high probability, we may consider the typical case to be when all participants follow the protocol exactly except for non-participation. In this case the round complexity is 4, plus at most 3 correction steps. In terms of message complexity, we transmit $O(M^2) = O(k^2)$ messages for every anonymous message sent. The bit complexity per anonymous bit sent is $O(k^4)$ in the worst case. Because k is unrelated to n, the number of participants, this protocol scales very well.

In the case where $O(k)$ parties send anonymous messages per round, the Transmission Phase of Protocol 3 still transmits $O(k^2)$ protocol messages for every anonymous message sent. However, there are alternate strategies that allow amortizing message complexity over the anonymous messages of the group.

One alternative is to replace this transmission phase by another in which, for each t such that $X[t] \neq 0$, each P_i randomly chooses $\lceil \frac{c}{1-\beta} \rceil$ members of $G[t]$ and sends $Msg[t]$ to those parties. In this case, when $O(k)$ parties transmit anonymously the ratio of protocol messages to anonymous messages is $O(k)$, and the ratio of protocol bits to anonymous bits is $O(k^3)$. However, *all* of the honest parties of the sending group fail to send to the *intended* recipient of $Msg[t]$ with probability $e^{-c/2}$. This condition is undetectable by the anonymous sender, thus requiring forward erasure correction over message blocks.

Another alternative trades round complexity for message complexity in the "best case": After each P_i in the sending group computes $X[t]$, P_i sends $Msg[t]$ to the i^{th} member Q_i of $G[t]$. Each Q_i then sends P_i a signature on $Msg[t]$. Finally each P_i collects all such signatures and broadcasts these signatures to the other members of his group. In this alternative, the round complexity increases by 2, but again when $O(k)$ anonymous messages are transmitted the ratio of protocol messages to anonymous messages is $O(k)$ and the ratio of protocol bits to anonymous bits is $O(k^3)$; and any member of the sending group who fails to forward anonymous messages is caught. However, this alternative is not secure against non-participation.

We intend for our protocol to be used over the Internet or networks of similar characteristics. Our protocol is particularly efficient in such networks, since throughput is frequently constrained by network latency, and our protocol has low round complexity.

Notice also that the zero-knowledge subprotocol is very efficient: with security parameter λ (the number of parallel repetitions of Protocol 4), the number of rounds is constant, the total number of bits transmitted is $O(k\lambda \lg p)$, and a non-conforming party is caught with probability at least $1 - 2^{-\lambda}$. However, even if it were less efficient, since it need only be executed when cheating takes place, and all cheaters can be caught with high probability, the cost of detection when amortized over many rounds is essentially zero.

5 Network Construction

The protocols in the previous section work for any network which has already been partitioned into groups. Here we present several strategies related to the efficient, scalable construction and management of this group structure.

5.1 Group Size

We set the group size to $M = \frac{2k}{1-\beta}$ (recall that β is the fraction of participants that the adversary can control) so that when the groups are chosen at random, with high probability at least k members of every group are honest: a multiplicative Chernoff Bound tells us that for any group G,

$$\Pr[|H \cap G| < k] \leq e^{-k/4} ,$$

so the probability that any group does not maintain k-anonymity decreases exponentially with k. For small k this probability can be computed directly for a tighter bound.

5.2 Group Formation and Management

We propose that a simple protocol be used to construct the groups. The formation of groups should be such that parties cannot choose which group they belong to. In an initialization phase, interested parties may securely construct the list $\langle P_1, \ldots, P_n \rangle$ either through a small group of trusted registration servers or through a secure group membership protocol such as that of [15]. The parties then choose a session identity S, for example, using a cryptographic hash function H applied to the initial parameters of the network. The number of groups is determined as the largest power of 2 smaller than $n(1-\beta)/k$, say 2^m. Then each P_i determines their group number by the m least significant bits of $H(S||P_i)$; thus any party, given the list of participants, can determine the group of any other party, and the other participants in his own group.

5.3 Optimizations and Concerns

5.3.1 Minimizing Turnover

If a significant number of honest parties leave the network (even temporarily) then the k-anonymity property may sometimes be violated. A possible approach to minimize this risk is by charging a high computational cost to rejoin a group, using a protocol such as Dwork and Naor's moderately hard functions [8] or Back's Hashcash [1].

5.3.2 Rate Adjustment

Notice that a significant barrier to the implementation of a fully anonymous protocol such as DC-Nets is the need to fully synchronize n hosts when n is large. In the protocol proposed here, there is no such requirement — the groups may operate asynchronously of one another. Because of that, each individual group may optimize its time between rounds to approximate the average sending rate of the group. This can be accomplished automatically using the fact that the outcome of the protocol gives a good estimate of the number of parties transmitting each round; so if no parties transmit, an additive increase in the intra-round gap may be used, and if many parties transmit, a multiplicative decrease may be used, as in other fair communications protocols.

6 Conclusions

We have introduced the notion of k-anonymous message transmission by analogy to the concept of k-anonymity from the privacy literature. Using this notion we are able to give simple and efficient protocols for anonymous message transmission which have provable security against a very strong adversary. We believe an interesting avenue for further research is to investigate whether other multiparty computation tasks can also be simplified using a similar approach, i.e. by weakening the security goals in a manner which is still sufficient for many applications. We also believe an important future step is the implementation of our protocol in order to determine the actual overhead introduced and the achievable throughput.

Acknowledgements

This material is based upon work partially supported by the National Science Foundation under Grants CCR-0122581 and CCR-0058982 (The Aladdin Center). This work was also partially supported by the Army Research Office (ARO) and the Center for Computer and Communications

Security (C3S) at Carnegie Mellon University. Nicholas Hopper was also partially supported by a NSF graduate research fellowship. The authors wish to thank Manuel Blum, Bartosz Przydatek, Mike Reiter, Latanya Sweeney, and the anonymous CCS reviewers for helpful discussions and comments.

References

[1] Adam Back. Hashcash. Unpublished manuscript, May 1997. Available electronically at `http://www.cypherspace.org/hashcash/`.

[2] Mihir Bellare and Phil Rogaway. Random Oracles are Practical. *Computer and Communications Security: Proceedings of ACM CCS'93*, pages 62-73, 1993.

[3] Ted Bridis. *Verizon Loses Suit Over Music Downloading.* Associated Press, April 24, 2003.

[4] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology* 1(1), pages 65-75, 1988.

[5] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM* 24(2), pages 84-88, 1981.

[6] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing It. *Advances in Cryptology: CRYPTO'86*, pages 200-212, 1987.

[7] Danny Dolev, Cynthia Dwork, Orli Waarts and Moti Yung. Perfectly Secure Message Transmission. *Journal of the ACM* 40(1), pages 17-47, 1993.

[8] Cynthia Dwork and Moni Naor. Pricing via Processing, or: Combating Junk Mail. *Advances in Cryptology: CRYPTO'92*, pages 139-147, 1993.

[9] The GNUnet website. `http://www.ovmj.org/GNUnet/`.

[10] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest Verifier Statistical Zero-Knowledge Equals General Statistical Zero-Knowledge. *Proceedings of 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 399-408, May 1998.

[11] Shafi Goldwasser, Silvio Micali, Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). *Proceedings of 17th Annual ACM Symposium on Theory of Computing (STOC'85)*, pages 291-304, 1985.

[12] David Mazieres and M. Frans Kaashoek. The Design, Implementation, and Operation of an Email Pseudonym Server. *Computer and Communications Security: Proceedings of ACM CCS'98*, pages 27-36, 1998.

[13] Torben P. Pedersen. Non-Interactive and Information Theoretic Secure Verifiable Secret Sharing. *Advances in Cryptology: CRYPTO'91*, pages 129-140, 1991.

[14] Andreas Pfitzmann and Michael Waidner. Networks Without User Observability – design options. *Advances in Cryptology: EUROCRYPT'85*, pages 245-253, 1985.

[15] Michael K. Reiter. A secure group membership protocol. *IEEE Transactions on Software Engineering* 22(1), pages 31-42, 1996.

[16] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security* 1/1, pages 66-92, 1998.

[17] Emin Gün Sirer, Milo Polte, and Mark Robson. CliqueNet: A Self-Organizing, Scalable, Peer-to-Peer Anonymous Communication Substrate. Unpublished manuscript, December 2001. Available electronically at `http://www.cs.cornell.edu/People/egs/papers/cliquenet-iptp.pdf`.

[18] Latanya Sweeney. k-Anonymity: a Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 10(5), pages 557-570, 2002.

[19] Michael Waidner. Unconditional sender and recipient untraceability in spite of active attacks. *Advances in Cryptology: EUROCRYPT'89*, pages 302-319, 1989.

[20] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. *Proceedings of ISOC Symposium on Network and Distributed System Security*, 2002.

A Protocol 2 does not need Reliable Broadcast

One technical point is not addressed in our presentation: since we avoid the use of reliable broadcast, is it possible for an adversary to disrupt the protocol by sending different messages to different parties in place of broadcasts? It is intuitively clear that the commitments used in the multiparty sum protocol (Protocol 2) prevent this situation as long as all parties participate; but since we are not aware of a published proof to this effect, we outline one here. The idea of the proof is conceptually simple: first we show that no single adversarial party may force an inconsistent outcome, and then we show that any set of k adversarially controlled parties can be successfully simulated by a single party. The result follows.

Lemma 1. *For any n, if discrete logarithms in $\mathbb{Z}_p^*$ are hard, no single party may cause two honest parties to compute different outputs in Protocol 2 .*

Proof. There are only two opportunities for the adversary (without loss of generality, P_1) to cheat via the lack of reliable broadcast: he may send (at least) two different commitment vectors $\{C_{1,j} : j \in [n]\}$ in step 1, or he may send two different sum values (R_1, S_1) in step 3. Note first, that if any attempt at step 1 will be caught, then the adversary is constrained by the commitment protocol in step 3. Thus it remains to prove that any attempt to send two distinct commitment vectors $\{C^*_{1,j}\}, \{C^\dagger_{1,j}\}$ is subsequently caught. To see that this is true, notice that there must be some j, such that $C^*_{1,j} \neq C^\dagger_{1,j}$. Without loss of generality, suppose $j = 2$. Furthermore, at least one party must receive C^*, and at least one must receive $C^\dagger$, without loss of generality, suppose these parties are P_2 and P_3, respectively. Now, suppose that P_1 incorrectly opens $C^*_{1,2}$ to P_2; obviously this is caught by P_2 in step 2. Otherwise, suppose P_1 correctly opens $C^*_{1,2}$ to P_2; then when P_3 receives the value $(R_2, S_2) = (r^*_{1,2} + \sum_{j\geq 2} r_{j,2}, s^*_{1,2} + \sum_{j\geq 2} s_{j,2})$ from P_2, and checks if $g^{S_2}h^{R_2} = C^\dagger_{1,2}\prod_{j\geq 2} C_{j,2}$, this check will fail since $C^*_{1,2} \neq C^\dagger_{1,2}$, by assumption. □

Lemma 2. *Any group of k (out of n) adversaries who cause two honest parties to compute different outputs in Protocol 2 with significant probability may be simulated by a single adversarial party (out of $n - k + 1$) with the same success probability.*

Proof. Without loss of generality, denote the k adversarially controlled parties of the hypothesis by $P_{n-k+1}, \ldots, P_n$, and let $\ell = n-k$. We will show how a single adversarial party Q may simulate the interaction between the honest parties $P_1, \ldots, P_\ell$ and the adversarial parties. First, without loss of generality, assume the parties $P_{\ell+1}, \ldots, P_n$ to be *delaying adversaries*: that is, all adversarially controlled parties wait until every honest party has spoken in each round. (If they do not wait, they can be rewritten to do so without decreasing their success probability) Then Q can simulate the honest parties to $P_\ell, \ldots, P_n$ as follows:

1. **Commitment Phase:** When P_i sends Q the commitment $C_{i,\ell+1}$, Q computes a random sharing of this commitment:
   - Q chooses k random shares $s'_{i,\ell+1}, \ldots, s'_{i,n}$ subject to $\sum_j s'_{i,\ell+j} = 0$.
   - Q chooses k random values $r'_{i,\ell+j} \in \mathbb{Z}_q$.
   - Q computes $C'_{i,\ell+1} = C_{i,\ell+1} h^{s'_{i,1}} g^{r'_{i,\ell+1}}$, and $C'_{i,\ell+j}$ $= g^{s'_{i,\ell+j}} h^{r'_{i,\ell+j}}$ for $2 \le j \le k$.
   - Q sends $\{C_{i,j} : j \le \ell\}, \{C'_{i,j} : \ell < j \le n\}$ to each adversarially controlled party.
2. **Sharing Phase:** when P_i sends Q the values $r_{i,\ell+1}$, $s_{i,\ell+1}$, Q sends $(r_{i,\ell+1} + r'_{i,\ell+1}, s_{i,\ell+1} + s'_{i,\ell+1})$ to $P_{\ell+1}$, and $(r'_{i,\ell+j}, s'_{i,\ell+j})$ to $P_{\ell+j}$, for $2 \le j \le k$.
3. **Broadcast Phase:** When P_i sends (R_i, S_i) to Q, Q sends (R_i, S_i) to each $P_{\ell+j}$.

Notice that by following this procedure, Q perfectly simulates the honest parties to the adversarial parties. In the opposite direction, Q emulates $P_{\ell+1}, \ldots, P_n$ to the honest parties as follows:

1. **Commitment Phase:** If each $P_{\ell+i}$ sends the commitment vector $\{C^j_{\ell+i,l}\}$ to P_j, then Q sends the commitment vector $\{\prod_i C^j_{\ell+i,l}\}$ to P_j.
2. **Sharing Phase:** If each $P_{\ell+i}$ sends the value $(r_{\ell+i,j},\, s_{\ell+i,j})$ to P_j, then Q sends the value $(\sum_i r_{\ell+i,j}, \sum_i s_{\ell_i,j})$ to P_j.
3. **Broadcast Phase:** If each $P_{\ell+i}$ sends the value (R^j_i, S^j_i) to P_j, then Q sends $(\sum_i R^j_i, \sum_i S^j_i)$ to P_j.

If the messages sent by $P_{\ell+1}, \ldots, P_n$ all pass all of the checks in Protocol 2, then so do the messages sent by Q. Thus Q forces an inconsistent outcome with the same probability as $P_{\ell+1}, \ldots, P_n$, as claimed. ☐

Theorem 6. *If discrete logarithms in $\mathbb{Z}_p^*$ are hard, no adversary may cause two honest parties to compute different outputs in Protocol 2 .*

Proof. The theorem follows by the conjunction of lemma 1 and lemma 2: since any k adversarial parties can force an inconsistent outcome with the same probability as some individual party, and no individual party may force an inconsistent outcome if discrete logarithms in $\mathbb{Z}_p^*$ are hard, then if discrete logarithms in $\mathbb{Z}_p^*$ are hard, no adversary (controlling any number of parties) may force an inconsistent outcome. ☐