Efficient Byzantine k-Anonymous Broadcast

Bryan Turner
bryan.turner@pobox.com
August, 2006

Abstract

Anonymous message transmission allows participants to exchange messages while keeping the sender and receiver identities private. Protocols based on the Dining Cryptographers problem are subject to jamming attacks by an adversary. We describe a novel technique to discover the identity of the adversary with no additional message complexity, and very low overhead.

Dining Cryptographers

The Dining Cryptographers Problem was introduced by David Chaum in 1988 [1] as a technique for sender and recipient untracability. The technique was generalized to Secure Multiparty Sums and forms the basis of Anonymous Broadcast channels called DC-Nets.

Secure Multiparty Sums

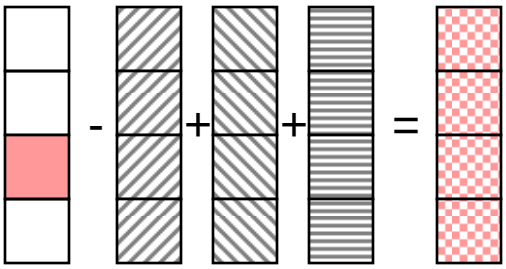
In Secure Multiparty Sum protocols for *N* parties, each participant selects a message to broadcast each round. A participant with no message to broadcast selects the message Zero. Broadcast is divided into slots, such that *N* slots exist in each round. Collisions occur if two parties attempt to broadcast in the same slot on any given round.

Phase 1:

The participants generate *N*-1 random strings each equal in length to a full set of *N* slots. The message to be broadcast is inserted into the correct slot of a zeroed string. Finally the ciphertext message is calculated – the plaintext message minus the sum of the random messages.

P = Plaintext of message to broadcast this round
R_i = Random message i
C = Cyphertext
O = Set of outgoing messages

$C = P - \Sigma_i R_i$
$O = \{ R_1 \ldots R_{N-1}, C \}$



$$P - (R_1+R_2+R_3) = C$$

Equation for the 4-party scenario.

Each participant exchanges one random outgoing message with each other participant, retaining one of the messages locally. In return, it receives one message from each other participant.

Phase 2:

Each participant computes the sum of all messages received and the locally-retained message:

M_i = Message received from participant i
L = Locally-retained message from Phase 1

$$S = L + \Sigma_i M_i$$

Each participant sends the calculated sum to each other participant, receiving in return the sums calculated by others. Finally, the sums are combined, producing the output for the round:

O = Output message
S_i = Sum from participant i

$$O = \Sigma_i S_i$$

The output contains a message from each participant, one per slot. The protocol is sender and receiver anonymous, but suffers from jamming. If an adversary sends messages in each slot, all messages will be corrupted – and the adversary is protected due to the sender-anonymous nature of the channel!

Divide & Conquer

In order to discover the identity of an adversary jamming the channel, we may introduce commitments as in [2,3]. This increases the computational and message complexity of the protocol. Developing an efficient alternative which reduces the complexity is the goal of this paper.

Intuitively, if the adversary cannot be singled out from one anonymous broadcast channel, it may be possible to arrange a series of channels in which the rogue would be revealed. For instance, in a 4-party scenario, it is possible to arrange the parties into 4 channels each containing 3 members such that each member is excluded from one of the channels (see figure 1). It then becomes trivial to oust the jammer; of the four channels, the one in which he is not present will be free from jamming.

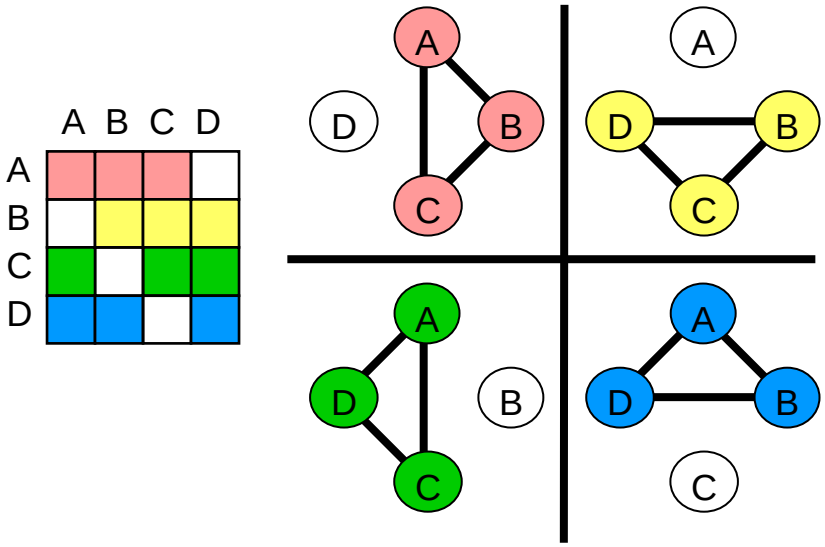


Figure 1. Left: Connection matrix for 4-party scenario. Right: Connection graph.

Details of the 4-Party Scenario

In order to discuss the later generalization, we will first examine the simplest scenario in detail. Note that 3 party anonymous channels cannot be designed using this technique, as they lack sender anonymity in their sub-channels.

Figure 1 illustrates the connection matrix and graph associated with the four broadcast channels. Each channel includes a subset of the nodes in the group such that one member is excluded from each group. Honest participants will not join a broadcast group that does not meet this connection matrix. Because of this, it is trivial to prove that all members will obey the connectivity matrix, including the adversaries.

In each round, the broadcast sub-channels provide one transmission slot per member (in this case, 3 slots per sub-channel). Each participant selects one slot per sub-channel to broadcast in and the protocol for each sub-channel is the standard multiparty sum protocol. We will assume a reservation protocol exists for these slots which enables honest participants to avoid colliding. A reservation protocol is described in later sections.

The output of the protocol is the complete set of messages broadcast in each slot. Note that any participant can jam at most N-1 of the N channels. Assuming honest participants never jam a channel, the adversary is trivially revealed.

Intelligent Adversaries

An intelligent adversary may decide to only jam one sub-channel, or to alternate among the sub-channels in which it participates. However, in jamming any round he reveals his presence, and his membership in that sub-channel. Additionally, the protocol continues to output messages even while being jammed.

Therefore the adversary must weigh the tradeoffs; by jamming more sub-channels he hinders the protocol, but his identity is closer to being revealed. While jamming fewer channels maintains his anonymity, but does little to stall the protocol.

Protocol Efficiency

The protocol efficiency can be improved from a naïve implementation. Each participant groups phase 1 and phase 2 messages by destination, sending all data in one message exchange per phase per participant. Thus the message complexity is equivalent to a single-channel secure multiparty sum protocol.

The protocol for an N-party scenario consists of each participant exchanging two phases of messages with each other participant, a total of 2N(N-1) discrete messages for each complete round (equivalent to the standard secure multiparty sum protocol).

Message size complexity per exchange consists of $(N-1)^2$ slots of data per participant of which (N-1) contain messages from the participant. A total of $2N(N-1)^2$ slots are exchanged each round. Interestingly, this is also equivalent to the standard secure multiparty sum. Intuitively, the standard protocol exchanges 2N(N-1) slots per round, where each participant has one output slot. While the new protocol exchanges $2N(N-1)^2$ of which (N-1) slots are available to each participant – which equates to 2N(N-1) slot exchanges per output slot.

Generalizing for Additional Adversaries

It is natural to examine an extension to the protocol to achieve protection from *f*-Byzantine faults. Intuitively, this requires arranging a connectivity matrix such that each sub-channel is missing *f* participants. Byzantine protection requires $N = 3f+1$ participants [5]. Examples of 2- and 3-Byzantine connectivity graphs are illustrated in Figure 2.

Protocol details follow trivially.

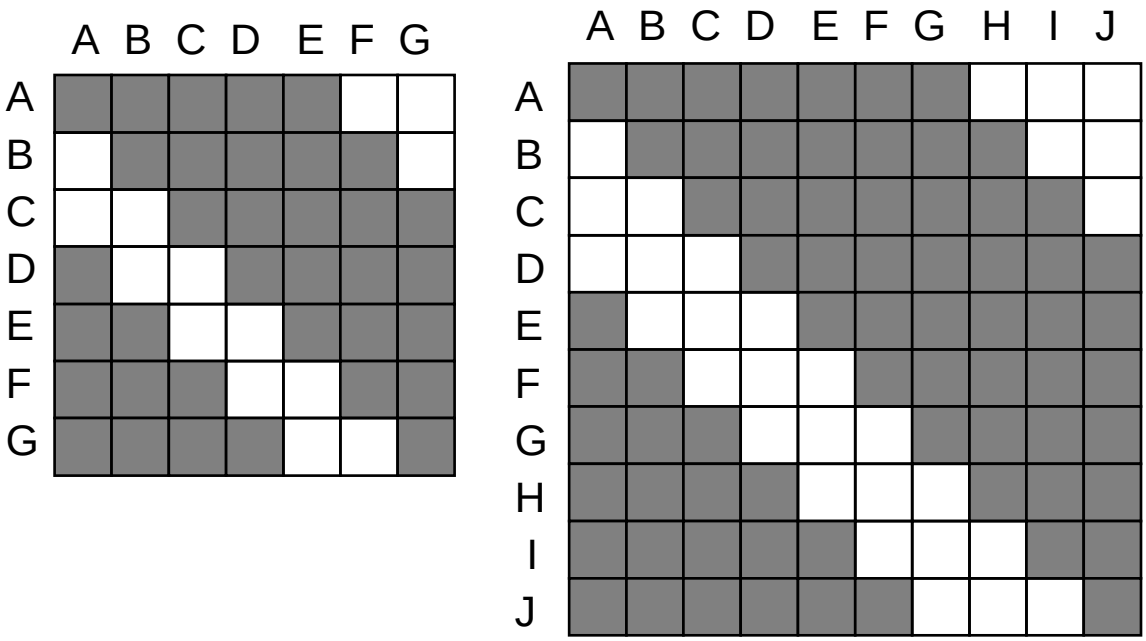


Figure 2: 2- & 3-Byzantine connectivity graphs.

Message Integrity

In order to prove the identity of a jamming adversary, all participants must agree. As only the broadcasting participant knows if its message was corrupted in the output, it is impossible for the other participants to learn of the attack.

To solve this, we define each broadcast slot to contain a message, plus a message integrity code (such as a CRC or HMAC). Each honest participant checks all output messages' integrity and unanimously eject jammers from the group.

Anonymous Block Reservation Protocol

In order to avoid collisions, honest participants may reserve slots in one round for broadcast slots in the next round. Each sub-channel defines a header of N^2 bits. Each participant randomly selects a bit and sets it to 1. The header is appended to the front of the message slots for the current round, and is secured in the same manner as all output messages.

At the end of the round, the output will include a header with one active bit per participant. Collisions among reservation bits is small, on the order of 1 collision per (N^2-N+1) rounds. For the 4-party scenario this is 1 reservation collision in 13 rounds.

The bits in the header define the order of broadcast slots; the participant which selected the lowest-order active bit broadcasts in the first slot, the participant which selected the second-lowest-order active bit broadcasts in the second slot, and so on.

In rounds with collisions each participant broadcasts in the appropriate slot, as before, but one slot will be corrupted by the collision. As each node can check the integrity of all output messages, honest participants can verify that only one message was corrupted in this round. An adversary which corrupts more slots is revealed using the techniques described previously.

Future Research

Multiparty Sum protocols have many similarities to Network Coding. Recent research [4] in Network Coding has solved jamming attacks by incorporating redundancy into the code. Cross-polinization of these research areas may prove fruitful.

Byzantine multiparty sums require exponentially increasing memory requirements in the number of participants. It may be possible to utilize probabilistic techniques to design connectivity graphs which reveal adversaries over time, with a small probability per round. Such techniques may allow linear memory requirements in the number of participants.

References:

[1] **The Dining Cryptographers Problem**
David Chaum
http://www.ece.cmu.edu/~adrian/731-sp04/readings/dcnets.html

[2] **k-Anonymous Message Transmission**
Luis von Ahn, et. al.
http://crypto.stanford.edu/~abortz/work/k-anon-final.html

[3] **A New k-Anonymous Message Transmission Protocol**
Gang Yao, Dengguo Feng
http://dasan.sejong.ac.kr/~wisa04/ppt/9A2.pdf#search=%22k-Anonymous%20Message%20Transmission%22

[4] **Resilient Network Coding in the Presence of Byzantine Adversaries**
S. Jaggi et. al.
http://pubs.jaggi.name/ncerror_infocom.ps.gz

[5] **The Byzantine Generals Problem**
L. Lamport, et. al.
http://research.microsoft.com/users/lamport/pubs/pubs.html#byz