Towards Provably-Secure Scalable Anonymous Broadcast

Mahdi Zamani
University of New Mexico
Albuquerque, NM
zamani@cs.unm.edu

Jared Saia
University of New Mexico
Albuquerque, NM
saia@cs.unm.edu

Mahnush Movahedi
University of New Mexico
Albuquerque, NM
movahedi@cs.unm.edu

Joud Khoury
Raytheon BBN Technologies
Cambridge, MA
jkhoury@bbn.com

Abstract

We devise a scalable and provably-secure protocol for fully-anonymous broadcast in large-scale networks. Similar to the dining cryptographers networks (DC-NETS), our algorithm uses secure multi-party computation (MPC) to ensure anonymity. However, we address the weaknesses of DC-NETS, which are poor scalability and vulnerability to jamming attacks. When compared to the state-of-the-art, our protocol reduces the total bit complexity from $O(n^2)$ to $\tilde{O}(n)$ per anonymous message sent in a network of size n; total latency increases from $O(1)$ to $\mathsf{polylog}(n)$. We assume up to a one third fraction of the parties is controlled by a static Byzantine adversary. We further assume that this adversary is computationally unbounded, and thus make no cryptographic hardness assumptions.

1 Introduction

Anonymous communication allows individuals to communicate with each other without fear of surveillance. An anonymity system attempts to conceal the relation between messages and their intended recipients, between messages and their actual senders, or both (*full anonymity*).

Today, political and commercial entities are increasingly engaging in sophisticated cyber-warfare to damage, disrupt, or censor information content [11]. In designing anonymity systems, there is a need to ensure reliability even against a powerful adversary. In this paper, we consider a Byzantine adversary that controls up to a certain fraction of parties in the network. The controlled parties can launch both passive (*e.g.* eavesdropping, non-participation) and active attacks (*e.g.* jamming, message dropping, corruption, and forging).

Two widely-accepted architectures for providing anonymity against such an adversary are Mix networks (MIX-NETS) and Dining Cryptographers networks (DC-NETS), both of which were originally proposed by Chaum [1, 2]. MIX-NETS are cryptographic in nature, require semi-trusted infrastructure nodes, and are known to be vulnerable to traffic analysis and active attacks [12].

DC-NETS [2, 9, 13, 14], on the other hand, provide unconditionally-secure anonymous broadcast protocols among a group of parties without requiring trusted third-parties. The core idea of DC-NETS is that a protocol for secure multi-party computation (MPC) can be used to perform sender and receiver anonymous broadcast. For example, if party p_i wants to broadcast a message m_i anonymously, then all other parties participate in a secure multi-party sum with input zero, while party p_i participates with input m_i. All parties learn the sum, which is m_i while all inputs remain private. This ensures that no party can trace the output message m_i to its input, keeping p_i anonymous.

Although DC-NETS are provably-secure against traffic analysis, they face several challenges. First, a reservation mechanism is required to schedule which party is broadcasting without compromising the anonymity of the sender. Second, DC-NETS are susceptible to collisions, which degrade throughput. A jamming adversary may even use collisions to render the channel useless by continuously transmitting in every round. Third, typical DC-NETS are not scalable given that the bit complexity required to anonymously broadcast a single bit within a group of size n is $\Omega(n^2)$.

State-of-the-art approaches that address some of these challenges include [4, 9, 13]. The majority of these methods are cryptographic in nature, and scale poorly with network size, rendering them impractical for large networks. We are not aware of any unconditionally-secure anonymous protocol that scales better than $O(n^2)$ bits per anonymous bit sent.

In this paper, we address the scalability and jamming limitations of DC-NETS. We do so using the recent scalable MPC algorithm of [5]. Our jamming-resistant protocol provides full anonymity at a total bit complexity

of $\tilde{O}(n)$ per anonymous message[1] and a total latency of $\mathsf{polylog}(n)$.

Our protocol is *provably-secure* as it is based on a formal security framework, which follows from the security of secure multi-party computation. In particular, our protocol is not vulnerable to end-to-end traffic analysis attacks common to circuit-based approaches like Tor [7]. We also provide provable anonymity against *a priori* knowledge that an adversary might have regarding the potential communicating parties, another property that Tor fails to provide[2]. Moreover, unlike Tor, which relies on centralized path setup servers, our protocol is fully-decentralized.

Our result is motivated by a vision of creating peer-to-peer versions of microblogging services with large number of users such as Twitter, but with provable anonymity guarantees. In Twitter, users can tolerate a higher messaging latency when compared to interactive web browsing applications. Therefore, trading-off latency for bandwidth cost and load-balancing is a promising goal for such applications.

2 Model and Problem Statement

We assume a network of n parties whose identities are common knowledge[3]. We assume there is a private and authenticated communication channel between every pair of parties and the communication is *synchronous*. Moreover, our protocol does not require the presence of any trusted third-party and we do not assume the existence of a reliable broadcast channel.

We assume $t < (1/3 - \varepsilon)n$ of the parties are controlled by a Byzantine adversary, for some positive constant ε. The adversary is actively trying to prevent the protocol from succeeding by attacking (1) the anonymity of the senders and receivers, and (2) the integrity of communications, by attempting to corrupt, forge, or drop messages. We say that the parties controlled by the adversary are *dishonest* and that the remaining parties are *honest*. The adversary is *computationally unbounded* thus, we make no cryptographic hardness assumptions. We also assume that the adversary is *static* meaning that it must select the set of dishonest parties at the start of the protocol. Finally, we make the standard assumption that the honest parties strictly follow our protocol, and do not form coalitions in which information is exchanged.

We say a network is *fully-anonymous* if it provides both sender and receiver anonymity. A protocol is *sender (receiver) anonymous* if the adversary is unable to distinguish between any of the honest parties as the sender (receiver) of a message[4]. More formally, a protocol is sender (receiver) anonymous if conditioned on all messages and information that the adversary gathers from all dishonest parties, the probability of an honest party being the actual sender (receiver) of a message is at most $1/(n-t)$.

In this paper, we design a fully-anonymous and scalable anonymity system. We assume all participating parties are publicly known, *i.e.* we are not trying to hide the participants' identities or their locations.

Note on the synchrony assumption The synchrony assumption in our model is due to the result of [5]. We point out that our protocol can also work in a fully-asynchronous setting using the result of [6]. In that setting, however, we can only guarantee that, conditioned on all information that is gathered by the adversary, the probability that a given honest party is the sender of a message is at most $1/(n-2t)$ instead of $1/(n-t)$. Also, t must be decreased to less than $(1/8-\varepsilon)n$ in the asynchronous setting, for some positive constant ε.

3 Our Results

Our main result is as follows.

Theorem 1. *Assume there are n parties in a fully-connected network with private channels, up to $t < n/3$ of which are controlled by an adversary, and each party has a constant-size message to broadcast. If all honest parties follow the protocol of section 6, then with high probability:*

1. *Each honest party broadcasts its message to all other honest parties with probability $1/k$, where $k > 1$ is a constant,*
2. *The communication is fully-anonymous,*
3. *Each party sends $\tilde{O}(n)$ bits and performs $\tilde{O}(n)$ computations,*
4. *The latency of the protocol is $\mathsf{polylog}(n)$.*

4 Further Related Work

Von Ahn et al. [13] develop a cryptographic broadcast protocol based on DC-NETS that is resistant to a static Byzantine adversary. A set of n parties with private inputs compute and share the sum of their inputs without revealing any parties' input. The authors introduce k-anonymity, which means no polynomial-time adversary may distinguish the sender/receiver of a message from among k honest senders/receivers. To achieve k-anonymity, they partition the set of parties into groups of size $M = O(k)$ and execute a multi-party sum protocol inside each group. The jamming detection mechanism is weak against an adversary who may waste valuable resources by adaptively filling up to M channels. In the case where n-anonymity is desired, the protocol requires

$O(n^3)$ messages to be sent per anonymous message and the total bit complexity is $O(n^4)$. The protocol has latency that is $O(1)$ on average when the number of broadcasts is large, but which can be $O(n)$ in worst case for a single broadcast. While we similarly use a secure sum computation, our protocol is non-cryptographic and handles both jamming detection and correction.

Golle and Juels [9] employ cryptographic proofs of correctness to solve the jamming problem in DC-NETS assuming a static Byzantine adversary. The protocol detects jamming with high probability in $O(1)$ rounds, requiring a communication and computation complexity of $O(n^2)$. Their protocol assumes the existence of a reliable broadcast and a centralized trusted authority for key management distribution.

The Verdict protocol of [10] (which is based on Dissent [3]) has a client-server architecture and uses verifiable DC-NETS, where participants use public-key cryptography to construct ciphertexts, and knowledge proofs to detect and exclude jamming parties before disruption. The protocol assumes the existence of a few highly-available servers, where at least one server is honest. All servers must be alive, however, for the protocol to work. An interesting aspect of Verdict is that it is robust to a large fraction of Byzantine parties (up to $n-2$). The paper demonstrates empirically that the system scales well with the number of clients, when the number of servers is fixed.

The Xor-trees approach of [8] extends DC-NETS to achieve $O(n)$ amortized communication complexity, which is optimal. In this protocol, only a single user is allowed to send at any one time in a Xor-tree. Hence, the protocol is subject to performance degradation due to collisions as the number of users increases. The protocol assumes the existence of a public-key infrastructure and a non-Byzantine polynomial-time adversary. The communication complexity of the protocol is $O(n^2t^2)$ bits in worst case, where t is the number of dishonest parties. The latency of the protocol is $O(n)$ in worst case. However, a sender may broadcast large payloads to amortize the costs. The amortized latency of the protocol is $O(1)$.

5 Preliminaries

In this section, we define standard terms used throughout the paper and then move to a recent result used in our protocol.

Notation An event occurs *with high probability*, if it occurs with probability at least $1-1/n^c$, for any $c>0$ and sufficiently large n. We assume all computations occur over a finite field $\mathbb{F}$.

Secure MPC We make critical use of a scalable protocol for secure MPC. Assume n parties in a fully-connected synchronous network who want to jointly compute any arbitrary function over their inputs while keeping their inputs private.

Dani *et al.* [5] describe an algorithm for solving MPC in this setting. They create groups of parties with logarithmic size called *quorums*. In each quorum, at least a 2/3 fraction of parties is honest. Let C be a circuit that computes the desired function, f, over n inputs. For each gate G in circuit C, a quorum Q_G is assigned to G. This quorum is used to compute the output of G. Their protocol ensures that the parties in Q_G all learn the sum of the output of G plus a mask value R_G selected uniformly at random from the field $\mathbb{F}$. Shares of R_G are held jointly by the parties in the quorum, but the value is unknown to any individual. Thus, no party learns any information about the output of G, but the parties together have enough information to provide the input for computation of the masked output of the next gate. This procedure is repeated to compute the values for the gates in the next layer of the circuit. At the top level of the circuit, the output of f is computed and is sent down to all parties through all-to-all communication between the quorums.

Let f be any function over n inputs, and C be a circuit that computes f. Let m be the number of gates in C and d be the depth of C. The following theorem gives the main result of [5].

Theorem 2. [5] *There exists a perfectly-secure protocol that can compute f with high probability while ensuring each party sends $\tilde{O}(\frac{n+m}{n}+\sqrt{n})$ messages and performs $\tilde{O}(\frac{n+m}{n}+\sqrt{n})$ computations. This protocol has total latency $O(d+\mathsf{polylog}(n))$.*

6 Our Protocol

Assume n parties $p_1,...,p_n$, where for $1 \le i \le n$, party p_i has a message m_i to broadcast to the network anonymously. Party p_i chooses a number $l \in [1,r]$ uniformly at random and forms a vector $X_i = [x_{ij}]$, where $x_{il} = m_i$ and $x_{ij} = 0$ (for all $1 \le j \le r$ and $j \ne l$). For some constant $k > 1$, each of the $r = kn$ positions in X_i is referred to as a *slot*. For simplicity, we assume r is an integer power of two. The parties then run the MPC algorithm of section 5 to compute a function $f(X_1, X_2, ..., X_n)$ such that every party learns the vector addition $\sum_{i=1}^{n} X_i$ and none of the parties can send more than one non-zero input. In the following, we explain the circuit that computes function f.

Our Circuit Figure 1 shows the circuit, which consists of two major subcircuits: JAMDETECTOR that detects jamming inputs and ADDER that computes the component-wise addition. We now describe each part of the circuit in detail.

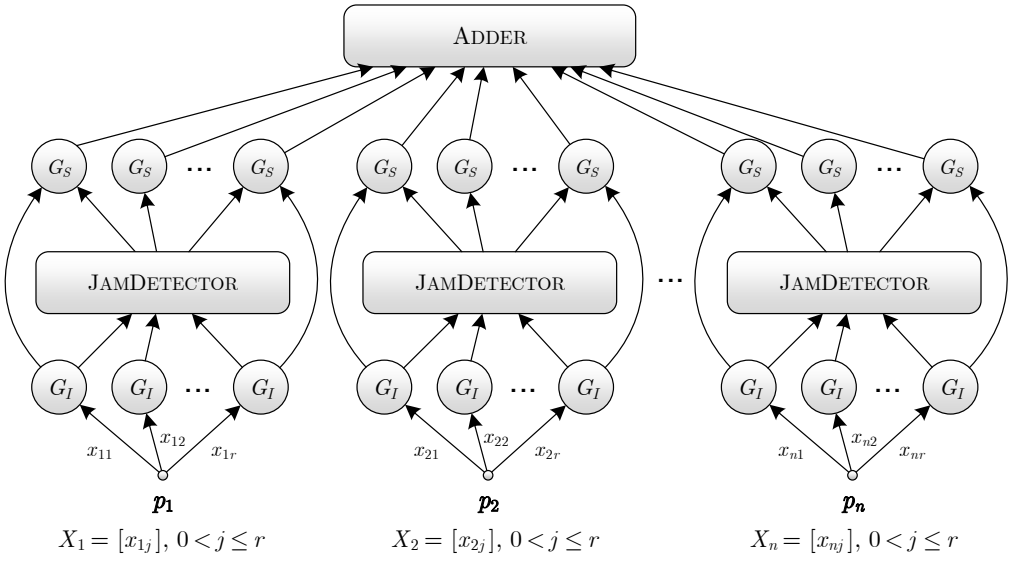

Figure 1: Our circuit for n parties showing JAMDETECTOR and ADDER subcirciuts as black-boxes.

- **Input gates:** In Figure 1, gates labeled G_I are called *input gates* and compute the identity function $G_I(x_{ij}) = x_{ij}$. Input gates are necessary for ensuring consistency among all inputs a party sends during the protocol execution.

- **Jam detector:** Each party is associated with exactly one JAMDETECTOR subcircuit. The subcircuit has r inputs and r outputs: all outputs are set to zero if no jamming is detected for the corresponding party otherwise all outputs are set to a non-zero value. Figure 2 depicts a circuit for $n = 2$ and $r = 4$ showing the subcircuit in detail. Each JAMDETECTOR consists of three types of gates, which are defined in the following on the field $\mathbb{F}$:

$$G_1(y) = \begin{cases} 0, & \text{if } y = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$G_2(y_1, y_2) = \begin{cases} 0, & \text{if } y_1 + y_2 = 0 \\ 1, & \text{if } y_1 + y_2 = 1 \\ 2 & \text{otherwise} \end{cases}$$

$$G_3(y) = \begin{cases} 0, & \text{if } y = 0, 1 \\ -1 & \text{otherwise} \end{cases}$$

  Each JAMDETECTOR contains a perfect binary tree consisting of only G_2 gates over r leaf nodes, consisting of only G_2 gates. This tree is connected from its root gate to an inverted perfect binary tree of only G_3 gates over $r/2$ leaf nodes, consisting of only G_3 gates.

- **Selector gates:** Gates labeled G_S in Figure 1 and Figure 2 are called *selector gates*. Each of these gates acts like a selector function: if the first input (which is an output of a JAMDETECTOR subcircuit) is zero, it simply outputs the second input otherwise it outputs zero. The selector gate is defined as follows:

$$G_S(y_1, y_2) = \begin{cases} y_2, & \text{if } y_1 = 0 \\ 0 & \text{otherwise} \end{cases}$$

  where y_1 is the output of corresponding G_3 and y_2 is the output of corresponding input gate.

- **Adder:** There is one ADDER subcircuit, which is a simple sum circuit and consists of r perfect binary trees each of which has n leaf nodes. The j-th binary tree sums up the outputs of all the j-th selector gates from all parties.

Circuit Computation Using the MPC algorithm described in section 5, the inputs of all parties are sent up the circuit simultaneously. The JAMDETECTOR subcircuit filters out any jamming inputs and sends the rest of them up to ADDER. The ADDER subcircuit computes the sum of all non-jamming inputs and finally, the result is sent down to every party via the output propagation algorithm described in [5]. The MPC algorithm ensures that (1) the output of the circuit is computed correctly and is reliably sent to all parties; and (2) no party learns any information about the inputs or outputs of intermediate gates, except what can be learned from their own input and the final output of the circuit.

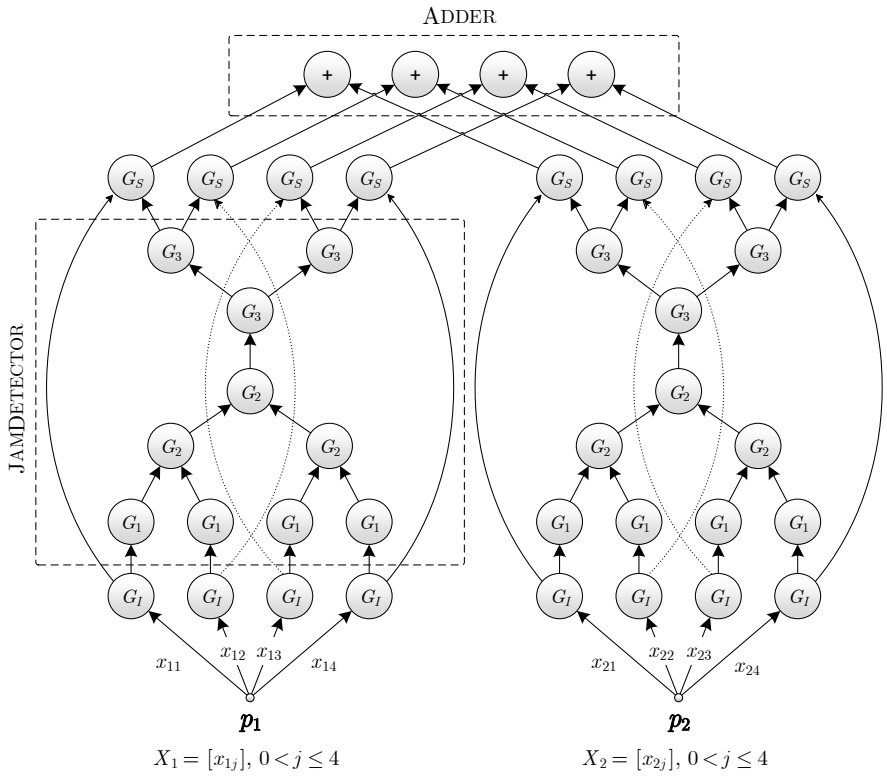

Figure 2: Our circuit for $n = 2$ and $r = 4$. The figure shows the JAMDETECTOR subcircuit in detail.

7 Analysis

In this section, we prove Theorem 1. As shown in the proof of Theorem 2 in [5], each gate in the circuit outputs the expected value.

Lemma 1. *Consider a* JAMDETECTOR *subcircuit, r input gates (G_I), and r selector gates (G_S) associated with party p_i (see Figure 1). During the computation of the circuit, at most one of the selector gates, say G_S^*, outputs a non-zero value. If p_i is honest, then G_S^* is chosen uniformly at random from the r selector gates.*

Proof. Assume p_i has a message m_i to broadcast. First, consider the case where p_i is honest. p_i chooses an input gate uniformly at random and sends m_i to that gate and sends `0` to all other input gates. Hence, at most one G_1 gate outputs `1` and the rest output `0`. Thus, all G_2 gates output `0` or `1`, and consequently all G_3 gates output `0`. Finally, the output of all G_s gates is the same as the output of the corresponding input gate. Hence, at most one G_s gate, say G_S^*, outputs a non-zero value. Obviously, G_S^* is chosen uniformly at random because the corresponding input gate is chosen uniformly at random.

Now, consider the case where p_i is dishonest and sends non-zero values to more than one of his input gates (this corresponds to a jamming attack). In this case, more than one G_1 outputs `1` and since the sum of them is greater than one, all G_3 gates output -1. Therefore, all G_s gates output `0`. □

In the protocol of section 6, we explained that each party selects one of the r slots uniformly at random. Even if all parties are honest (*i.e.* no jamming is occurring), *collisions* are always possible meaning that one or more parties may choose the same slot for their non-zero message. Unfortunately, there is no efficient non-interactive method that guarantees all parties select distinct slots. Hence, for simplicity, we ensure that the number of slots is large enough so that the probability of collision remains small. The following lemma gives an upper-bound on the probability of collisions.

Lemma 2. *If $r = kn$, where r is the number of slots and $k > 1$ is a constant, then the probability of collision for one party is less than $1/k$.*

Proof. By Lemma 1, we can ensure that each party sends his input to at most one slot, *i.e.* jamming has already been prevented. Let p be an arbitrary honest party. The probability that another party chooses exactly the same slot that p chooses is $1/r$. So, the probability of collision for p is at most $(n-1)/r$. Since $r = kn$, the probability of

collision is at most $(n-1)/kn < 1/k$. As in [13], in most cases we can set $r = 2n$, which makes the probability of collision for a party less than $1/2$. □

Lemma 3. *Our protocol is fully-anonymous and it ensures that for each honest party p_i, who sends message m_i, all honest parties learn m_i with probability $1 - 1/k$.*

Proof. Theorem 2 guarantees that the adversary only learns the set of outputs and the corresponding slots besides his own input. The input slots were chosen independently and uniformly at randomly by each party. As proved in Lemma 1, every m_j sent by an honest party p_j $(1 \leq j \leq n)$ appears in at most one of the corresponding selector gates uniformly at random and all other selector gates output `0`. Since the ADDER subcircuit keeps the ordering of slots, the output slots corresponding to honest parties are also chosen uniformly at random and thus, give the adversary no extra information. Therefore, conditioned on all information that the adversary can learn during the protocol, the probability that m_i is sent by any honest party p_j $(1 \leq j \leq n)$ is $1/(n-t)$. This means that the communication is sender-anonymous.

Theorem 2 guarantees that all honest parties learn every output of the circuit with high probability, one of which is m_i. This shows that the communication is receiver-anonymous and thus, the protocol is fully-anonymous. Moreover, based on Lemma 2, the probability that the protocol fails to deliver m_i is less than $1/k$. □

Lemma 4. *The circuit constructed in section 6 has depth $O(\log n)$ and $O(nr)$ gates, where r is the number of slots.*

Proof. In the circuit of Figure 1, for each party a subtree of depth $2\log r + 3$ consisting of $5r - 2$ gates is required ignoring the ADDER circuit, which consists of r binary trees each with n leaves. Each tree has depth $\log n + 1$ and consists of $2n - 1$ gates so the ADDER subcircuit has $(2n-1)r$ gates. Therefore, the total number of gates in a circuit for n parties is $7nr - 2n - r$ and the circuit has depth $2\log r + \log n + 4$. □

Lemma 5. *If all honest parties follow our protocol, then with high probability, the protocol sends $\tilde{O}(n)$ bits and performs $\tilde{O}(n)$ computations for sending one anonymous bit. The latency of the protocol is* $\mathsf{polylog}(n)$.

Proof. The MPC protocol of [5] requires each party to send $\tilde{O}(\frac{n+m}{n} + \sqrt{n})$ bits and perform $\tilde{O}(\frac{n+m}{n} + \sqrt{n})$ computations to compute any arbitrary function f, where m is the number of gates in the circuit for computing f. However, unlike [5] that propagates only one message (the result of computation) to every party at the output propagation phase, our protocol transmits n messages at this phase. Therefore, it requires each party to send $\tilde{O}(\frac{n^2+m}{n} + \sqrt{n})$ bits. From Lemma 4, we have $m = O(n^2)$ gates so each party sends $\tilde{O}(n + \sqrt{n}) = \tilde{O}(n)$ bits for sending n anonymous bits. Alternatively, the protocol sends $\tilde{O}(n)$ bits for sending one anonymous bit. With a similar argument, the protocol requires each party to perform $\tilde{O}(n)$ computations.

As Theorem 2 shows, the MPC algorithm takes $O(d + \mathsf{polylog}(n))$ rounds to compute any arbitrary circuit with depth d, where n is the number of parties. Lemma 4 shows that the circuit of section 6 for n parties has depth $O(\log n)$, since $r = O(n)$. Therefore, the protocol takes $\mathsf{polylog}(n)$ rounds to send an anonymous bit. □

8 Conclusion and Open Problems

We described a Byzantine-resistant protocol for fully-anonymous broadcast in large-scale networks that has a communication complexity of $\tilde{O}(n)$ per anonymous bit and a polylogarithmic latency. To the best of our knowledge, this is the first result that ensures provable anonymity against an unbounded adversary that sends asymptotically less than $O(n^2)$ bits per anonymous bit.

Although the $\tilde{O}(n)$ message complexity of our protocol scales very well comparing to the state-of-the-art, the actual bandwidth cost is still very high. This is mainly due to the large constants in the communication complexity of the scalable MPC algorithm and the large number of gates in our circuit. It may be possible to decrease the message cost significantly in practice by using threshold cryptography to speed up Byzantine agreement [15], which is used repeatedly in the MPC algorithm for simulating a reliable broadcast channel.

One aspect of our DC-NET-based model is that the adversary can only try to corrupt the protocol by actively jamming the channels, which makes it possible to detect dishonest parties. We are interested in the average cost of our protocol when multiple broadcasts occur and thus, it may be possible to blacklist parties that exhibit adversarial behavior. We hope to create an algorithm that achieves amortized resource costs per anonymous message sent that is significantly better than our algorithm for a single-shot communication.

Our work (and most previous work) assumes the identities of parties is a common knowledge. This does not conform, for example, with the fully-distributed nature of peer-to-peer systems. We are interested to know if we can do better in this respect and still retain efficiency and provable security.

References

[1] CHAUM, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24*, 2 (Feb. 1981), 84–90.

[2] CHAUM, D. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology 1* (1988), 65–75. 10.1007/BF00206326.

[3] CORRIGAN-GIBBS, H., AND FORD, B. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and Communications Security* (New York, NY, USA, 2010), CCS'10, ACM, pp. 340–350.

[4] CORRIGAN-GIBBS, H., WOLINSKY, D. I., AND FORD, B. Dining in the Sunshine: Verifiable Anonymous Communication with Verdict. *ArXiv e-prints* (Sept. 2012).

[5] DANI, V., KING, V., MOVAHEDI, M., AND SAIA, J. Brief announcement: breaking the o(nm) bit barrier, secure multiparty computation with a static adversary. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing* (New York, NY, USA, 2012), PODC '12, ACM, pp. 227–228. Full version available at `http://arxiv.org/abs/1203.0289`.

[6] DANI, V., KING, V., MOVAHEDI, M., AND SAIA, J. Quorums quicken queries: Efficient asynchronous secure multiparty computation, 2013. Available at `http://cs.unm.edu/~saia/papers/mpc-asynch.pdf`.

[7] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium - Volume 13* (Berkeley, CA, USA, 2004), SSYM'04, USENIX Association, pp. 21–21.

[8] DOLEV, S., AND OSTROBSKY, R. Xor-trees for efficient anonymous multicast and reception. *ACM Trans. Inf. Syst. Secur. 3*, 2 (May 2000), 63–84.

[9] GOLLE, P., AND JUELS, A. Dining cryptographers revisited. In *Proceedings of Eurocrypt 2004* (May 2004).

[10] HENRY CORRIGAN-GIBBS, D. I. W., AND FORD, B. Proactively accountable anonymous messaging in verdict. In *Proceedings of the 22nd USENIX Security Symposium* (2013).

[11] KREPINEVICH, A. F. Cyber warfare: A nuclear option?, 2012. Center for Strategic and Budgetary Assessments, Washington, DC, USA, 2012.

[12] PFITZMANN, A., AND WAIDNER, M. Networks without user observability design options. In *Proc. of a workshop on the theory and application of cryptographic techniques on Advances in cryptology—EUROCRYPT '85* (New York, NY, USA, 1986), Springer-Verlag New York, Inc., pp. 245–253.

[13] VON AHN, L., BORTZ, A., AND HOPPER, N. J. *k*-anonymous message transmission. In *Proceedings of the 10th ACM conference on Computer and communications security* (New York, NY, USA, 2003), CCS '03, ACM, pp. 122–130.

[14] WAIDNER, M., AND PFITZMANN, B. The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure serviceability. In *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology* (New York, NY, USA, 1990), EUROCRYPT '89, Springer-Verlag New York, Inc., pp. 690–.

[15] YOUNG, M., KATE, A., GOLDBERG, I., AND KARSTEN, M. Practical robust communication in dhts tolerating a byzantine adversary. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems* (Washington, DC, USA, 2010), ICDCS '10, IEEE Computer Society, pp. 263–272.

Notes

[1]The symbol $\tilde{O}$, is used as a variant of the big-O notation that ignores logarithmic factors. Thus, $f(n) = \tilde{O}(g(n))$ means $f(n) = O(g(n)\log^k g(n))$ for some k.

[2]For example, the adversary might know beforehand that some group of dissidents are likely to be communicating and could redirect all its resources to prove this.

[3]This is required only because of the quorum-building method we use in our scheme as a black-box. This means that our algorithm can be used without such common knowledge if also the quorum-building method can work without that knowledge.

[4]Obviously, delivering a message to all parties guarantees receiver anonymity [12].