3P3: Strong Flexible Privacy for Broadcasts

David Mödinger and Franz J. Hauck
Institute of Distributed Systems, Ulm University
Email: {david.moedinger, franz.hauck}@uni-ulm.de

Abstract—**Privacy concerns have reached the mainstream discourse in society and already had a significant impact on research and technology. Cryptocurrencies have adopted many transaction-level privacy mechanisms to provide privacy in the persisted blockchain. Unfortunately, these are insufficient as network-level attacks can also provide privacy-breaking insights into transactions and their origins. We proposed k-Dining Cryptographers and topological methods as a basis for a privacy-preserving broadcast protocol. In this work, we present 3P3, a three-phase privacy-preserving broadcast protocol. We transformed our approach to a stronger attacker model so that it provides strong base privacy against global attackers and malicious nodes and additional privacy against common attackers, e.g., botnets. Further, we provide mechanisms to transmit almost arbitrarily long messages, reduce overhead for zero-message rounds, a more extensive analysis, and simulation results of our enhanced protocol. Our simulations show the dissemination of a message to all nodes within 1000ms in 99.9% of instances. These results hold for all network sizes, including networks of up to 10,000 participants. Bandwidth estimates also show practical applicability with usual group sizes of 10 to 30 participants.**

I. Introduction

Cryptocurrencies and blockchains have been popular for years in research and industry. This phenomenon went as far as many people of varying technical proficiency investing their money in cryptocurrencies hosted by blockchains. Researchers noticed that many blockchains contained sensitive data of their users in unprotected and unprotectable forms. Such data contains information on purchasing behaviour, credit balances, and how the money was acquired [1], [2].

Many new cryptographic applications came to life to protect this sensitive information. Approaches to achieve unlinkable payments include the use of ring signatures [3]–[5] and zero-knowledge proofs [6], [7]. Even already existing blockchains like Bitcoin were augmented with privacy-enhancing mechanisms [8], [9].

However, these systems focus solely on privacy on the persisted blockchain, as every user can acquire and inspect this data. This publicly available data has been intensively investigated even for strong privacy-protected blockchains such as zcash [10]. Some researchers investigated the dissemination of transactions within the peer-to-peer network [11], [12]. They recognised a lack of network privacy for blockchain systems, leading to the development of protocols to protect the network-layer participants. Dandelion [13] tackles some of the issues on the network layer for Bitcoin. Monero [3] adopted a variant of the onion routing protocol I2P [14] called Kovri.

We proposed a design for a privacy protocol [15] with parameters making it flexible for various environments. Our previous proposal lacked a strong attacker model and performance and parameter evaluations. Building on the ideas of this previous proposal, we propose 3P3, a three-phase privacy-preserving broadcast protocol. Compared to our previous publication, we provide several improvements, as well as additional contributions:

- Base privacy against strong attackers even in the presence of malicious entities.
- Enhanced protection against common attackers.
- Adaptable message lengths for transmission.
- Reduced overhead for zero-message rounds.
- Comparative evaluation showing practical performance in real-world scenarios.

Section II introduces the necessary background information about this paper, while Section III discusses related work. Section IV describes the 3P3 protocol for a privacy-preserving broadcast in detail. In Section V, we show that our protocol is robust in the presence of malicious nodes. In Section VI, we discuss the privacy properties of our protocol. Lastly, Section VII provides a performance evaluation of our protocol.

II. Background

In this section, we describe the expected scenario, i.e. blockchains, our assumptions and requirements for the network. All relevant notation is summarized in Table I.

Term	Meaning
$x =\sim \mathcal{U}(a,b)$	Chose a random real number uniformly with $x \in [a,b]$.
$i =\sim \mathcal{U}\{a,b\}$	Chose a random integer uniformly with $a \leq i \leq b$.
$E =\sim \mathcal{U}_n\{S\}$	Select n distinct elements from S uniformly at random.
$\mathbb{G} = (E,V)$	Graph $\mathbb{G}$ consisting of a set of vertices V and edges E.
G	All participants of a group.
$\mathcal{N}$	The set of neighbours chosen in our modified diffusion.
$\eta = \lvert\mathcal{N}\rvert$	Size of $\mathcal{N}$.
$\mathcal{H}(x)$	Function to compute the hash of element x.
process(m)	Call application logic to process the given message m.
$C_r(x)$.	Commitment on value x, blinded by random factor r.
$\{X\}_i$	Value X encrypted under the public key of i.

TABLE I
Notation used in this paper.

A. Dining Cryptographer Networks

The base protocol of the dining cryptographer network (DCN) by Chaum [16] allows for unconditional sender untraceability. In its more general form, the protocol allows a single participant to share a message in an unlinkable way. Given a fixed-length message, e.g., by a modulus for modular arithmetic, all participants i decide on their message m_i to share. If participants do not intend to send a message, they will prepare $m_i = 0$. Only one participant is allowed to share a message $m_i \neq 0$. Each participant i splits their message in random slices, so that $m_i = \sum_{j \neq i} s_{i,j}$ and shares $s_{i,j}$ with participant j. All participants combine every slice they receive and share the combination with all other participants. Therefore, all participants will receive an aggregate of all slices produced by every other participant and can calculate the aggregation of all messages, which corresponds to $\sum_i m_i$.

Unfortunately, DCNs do not scale well with the number of participants and can be prevented from making progress by malicious participants. Von Ahn et al. [17] built upon other improvments [18] and addresses these two weaknesses in their modification. They apply the dining cryptographer protocol to groups of a maximum size $k << n$ instead of the whole network. Further, they designate a set of message slots to transmit multiple messages at once. Lastly, they use commitments in cooperation with a blame protocol to identify malicious actors after detecting collisions. These additions result in a protocol with more overhead compared to basic DCNs but provide fairness and robustness in malicious environments. The protocol by von Ahn et al. is a point-to-point protocol: a participant sends the message and an identifier for the target group.

B. Network

The protocol we describe requires an underlying peer-to-peer network. We assume the network is equivalent to an undirected, connected graph with no loops and no multiple edges. Further, we assume that the network remains connected, even if all malicious or faulty nodes are removed. Otherwise, the network could be partitioned by malicious nodes, and our protocol would be unable to reach all nodes. Known properties of connectedness [19, Ch. 4] can consider a given rate of attackers for network creation; therefore, our assumption is warranted and not a substantial restriction on the underlying network.

Channels between nodes should be authenticated, encrypted and integrity protected. Modern cryptographic schemes, e.g., via mTLS, can easily realise such a channel. Further, we require a public-private key pair for encryption with shared public keys among the group as well as an agreed-upon ordering within the group, e.g., ordered by public keys.

We assume the network underlying our protocol takes care of essential network functions and management operations, e.g., re-establishing interrupted connections, group join and leave operations, authentication and encryptions. Connections within groups should only be used for group communication, i.e., the DCN. This restriction prevents topology learning attacks on the groups, i.e., who belongs to a given group.

III. Related Work

While we focus on the scenario of blockchain transactions, there may be other application domains for 3P3.

A. Dandelion

Dandelion [13] proposes a two-phase protocol for statistical spreading. Phase 1 spreads the transaction along a line graph. Each node along the line graph has a fixed probability of starting the second phase. Phase 2 uses a flood and prune broadcast to ensure delivery to all nodes. For an attacker to detect the originator with reasonable certainty, he needs to be the first node receiving the message from the originator. With known topology, many attackers could further improve estimates without being the first recipient.

Dandelion++ [20] is an improvement over the initial version. The iteration covers enhanced defences against graph learning and graph construction attacks, as well as intersection, selective non-participation, and partial deployment attacks.

B. Adaptive Diffusion

Adaptive diffusion [21] breaks the symmetry present in regular flood and prune broadcast by creating a virtual source token. The owner of the virtual source token either spreads the message or forwards the virtual source token. If the token is transferred, the new owner becomes the centre by balancing the tree. Although this approach is for cycle-free networks, it works well even for general networks [21]. Adaptive diffusion does not guarantee delivery of messages to all nodes, which can lead to unfairness in the context of blockchains. Lastly, a suitably powerful attacker can subvert the protocol by connecting to a large number of nodes, as a node informs all neighbours of new messages.

C. Dissent

Dissent [22], [23] provides high privacy guarantees with a small number of core servers as anonymity providers. It uses a round-based multi-phase protocol. Every participant announces the length of the message they want to transmit to enable variable-sized messages. The length announcements stay anonymous using a secure random shuffle of layer-wise encrypted values.

The last participant publishes the shuffled lengths. To transmit the data, the participants use a form of a DCN. The announcement phase [22] scales linearly in the number of group members, e.g., a group size of 8 to 12 participants incurs a latency of around 30 seconds.

D. Onion Routing Protocols

Onion routing protocols, e.g., Tor [24], provide a point-to-point communication abstraction. The general concept applies a set of nodes to chain traffic via these nodes. Each package sent through this chain is encrypted with as many layers as there are nodes. Each node removes one layer of this encryption and forwards it to the next node. Garlic routing, e.g., Kovri and I2P, is a variant of onion routing, bundling multiple messages together.

Onion routing generally does not consider a global passive attacker listening to the connections, but recent wide-area surveillance of the internet makes them seem realistic. As Tor provides a socket proxy abstraction, it can be used in addition to other systems as a defence-in-depth approach.

E. Others

There are many other privacy-oriented systems besides the ones we presented here. These include systems for different use cases such as voice-over-IP with Herd [25], as well as point-to-point communication schemes such as Rac [26]. While these systems provide valuable building blocks, e.g., cover traffic, and solutions to existing problems, we did not apply them for various reasons, e.g., high network load and increased latency.

IV. 3P3: Privacy Prevserving Broadcast Protocol

3P3 consists of three phases: a modified version of a DCN for strong privacy, a modified adaptive diffusion, and a flood and prune broadcast to ensure delivery, cf. Figure 1.

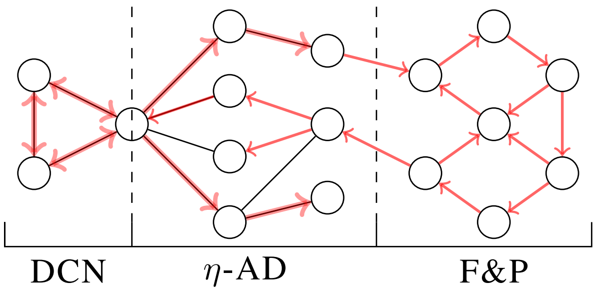

Fig. 1. Overview of the three phases of 3P3. A DCN, our modified η-diffusion (η-AD) and the final flood and prune (F&P) phase.

A. Phase I: Dining Cryptographer Network

In the first phase, we perform a preliminary round to establish all desired message lengths, similar to Dissent [23], using the algorithm by von Ahn et al. [17], shown by Algorithms 1 to 4. The actual messages are sent in a separate DC round.

Algorithm 1 Preparation phase of the first DC instance.

```
Input: Message $m$
Environment: Group $G$ with $|G| = k$, the executing node $g_{\text{self}}$, public keys of group participants, slot length $\ell$
  $r =\sim \mathcal{U}\{0, 2^{16}-1\}$
  Slot$=\sim \mathcal{U}\{0, 2k-1\}$
  $K = \{K_i = \{\sim \mathcal{U}\{0, 2^{256}-1\}\}_i\}$
  $X[i] = \begin{cases} (r, \text{length}(m), K) & \text{if } i = \text{Slot} \\ (0,0,0) & \text{else} \end{cases}$
  for $t \in \{1 \dots 2k\}$ do
    $s_{\text{self},i}[t] = \begin{cases} \sim \mathcal{U}\{0, 2^{\ell}-1\} & \text{if } i < k \\ X[t] \oplus \bigoplus_{j \in \{1 \dots k-1\}} s_{\text{self},j} & \text{if } i = k \end{cases}$
    $r_{\text{self},i}[t] =\sim \mathcal{U}\{0, 2^{\ell}-1\}$
    Compute $C_{\text{self},i}[t] = C_{r_{\text{self},i}[t]}(s_{\text{self},i}[t])$
  end for
  Broadcast $\{C_{\text{self},i}[t] : i \in \{1 \dots k\}, t \in \{1 \dots 2k\}\}$
```

The random identifiers r_{self} are required for a node to identify the slot of their message in the presence of multiple identical lengths. The birthday paradox formula provides a rough boundary, given a collision probability p and group size k:

$$\text{sizeof}(r) \geq \log_2\left(\frac{1}{1-(1-p)^{\frac{2}{k(k-1)}}}\right).$$

For a collision probability of 1%, we can tolerate 36 participants using 16 bits per random identifier. Collisions should already be rare due to requiring the same length. Most applications should function well with these values.

Algorithm 2 Sharing round of the DCN.

```
Input: $\forall i \in \{1 \dots k\} : s_{\text{self},i}, r_{\text{self},i}$ of algorithm 1
Environment: Group $G$, the executing node $g_{\text{self}}$, commitments $C_{i,j}[t]$
  for $g_i \in G \setminus \{g_{\text{self}}\}$ do
    Send $\{(r_{\text{self},i}[t], s_{\text{self},i}[t]) : t \in \{1 \dots 2k\}\}$ to $g_i : i \neq \text{self}$
    Receiving node $g_i$ validates that $C_{r_{\text{self},i}[t]}(s_{\text{self},i}[t]) = C_{\text{self},i}[t]$
  end for
```

Algorithm 3 Aggregation sharing phase of the DCN.

```
Input: $\forall i \in \{1 \dots k\} : (r_{j,\text{self}}[t], s_{j,\text{self}}[t])$ transmitted by others in algorithm 2
Environment: Group $G$, the executing node $g_{\text{self}}$, commitments $C_{i,j}[t]$
  Collect all $(r_{j,\text{self}}[t], s_{j,\text{self}}[t])$ pairs, sent to $g_{\text{self}}$
  $R_{\text{self}}[t] = \sum_j r_{j,\text{self}}[t]$
  $S_{\text{self}}[t] = \bigoplus_j s_{j,\text{self}}[t]$
  Broadcast $\{(R_{\text{self}}[t], S_{\text{self}}[t]) : t \in \{1 \dots 2k\}\}$
  Everyone checks $C_{R_{\text{self}}[t]}(S_{\text{self}}[t]) = \prod_j C_{j,\text{self}}[t]$
```

Algorithm 4 Result computation of the DCN.

```
Input: $\forall i \in \{1 \dots k\} : (R_j[t], S_j[t])$ broadcasted by others in algorithm 3
Environment: Group $G$, the executing node $g_{\text{self}}$, commitments $C_{i,j}[t]$
  Collect all $(R_j[t], s_j[t])$ pairs, sent to $g_{\text{self}}$
  $R[t] = \sum_j R_j[t]$
  $X[t] = \bigoplus_j S_j[t]$
  Everyone checks $C_{R[t]}(X[t]) = \prod_{i,j} C_{i,j}[t]$
  return $X[t]$
```

For a fixed group size of k participants, we prepare $2k$ slots for message lengths. This number is to ensure a probability of delivery greater than $\frac{1}{2}$ even when all participants send a message. All slots consist of a random identifier r, a message length ℓ and a set of k values K_i, encrypted to each participant i. When a participant i wants to disseminate a message m_i, they chose a slot and an identifier at random. Once this information is shared through the DCN, all nodes prepare a round with message length $2\sum_i \ell_i$, i.e., the sum of all provided lengths. The preparation is shown in Algorithm 5, while the transmission is identical to Algorithms 2 to 4, just based on the output of Algorithm 5.

Algorithm 5 Preparation phase of the second DC instance.

```
Input: Message $m$, $r$ from algorithm 1, $X$ from algorithm 4
Environment: Group $G$ with $|G| = k$, executing node $g_{\text{self}}$
  for $t \in \{1 \dots 2k\}$ do
    $(r[t], \ell[t], K[t]) = X[t]$
    $Y[t] = \begin{cases} m & \text{if } r[t] = r \\ 0 & \text{else} \end{cases}$
    $s_{\text{self},i}[t] = \begin{cases} \sim \mathcal{U}\{0, 2^\ell - 1\} & \text{if } i < k \\ Y[t] \oplus \bigoplus_{j \in \{1 \dots k-1\}} s_{\text{self},j} & \text{if } i = k \end{cases}$
    $r_{\text{self},i}[t] = \text{PRNG}_{\text{K[T][self]}}(\sim \mathcal{U}\{0, 2^\ell - 1\})$
    Compute $C_{\text{self},i}[t] = C_{r_{\text{self},i}[t]}(s_{\text{self},i}[t])$
  end for
  Broadcast $\{C_{\text{self},i}[t] : i \in \{1 \dots k\}, t \in \{1 \dots 2k\}\}$
```

Any node that submitted a length $\ell \neq 0$ can find their designated position by their position in the message slot. All other message parts are set to zero. The commitment aggregation to zero for these message parts needs to use the provided random value from round one. Using these prepared values, allows the author to blame any node that does not send a zero message in the author's slot. The messages are summarised in Table II.

If all lengths are zero, the second round can be skipped. Valid lengths should be restricted to prevent a denial-of-service attack, e.g., 16 bits for the use case of blockchains. While this leaks when something is sent, that is not a problem, as broadcasts could be traced to this group by a global passive attacker.

Round	Transmitted Message	Length
1.	$[(r_1, \ell_1, K_1), \dots, (r_{2k}, \ell_{2k}, K_{2k})]$	$2k(\text{sizeof}(r + \ell + K))$
2.	$[m_1, \dots, m_i]$	$2\sum_i \text{length}(m_i)$

TABLE II
PHASE 1 MESSAGES TRANSMITTING MESSAGES m_1 THROUGH m_i.

To assign responsibility to start the next round without communication, the node which has the node identifier closest to the random identifier should start the next round. Nodes sort the node identifiers and split the available space of values for random identifiers uniformly to prevent an uneven workload.

B. Phase II: Adaptive Diffusion

The second phase uses adaptive diffusion [21], which consists of two parts, the spreading mechanism and the virtual source subprotocol. The virtual source should forward messages so that all nodes of a given distance from them either all received the message or all did not receive the message.

If a node, which is not the virtual source, receives a message for the first time, they select η neighbours. Further, they store the node that sent the original message to prevent unwanted extension of the spreading sub-graph over cross-connections. Whenever they receive the same message again, they will forward the message to the selected neighbours. As adaptive diffusion was originally constructed for contact networks, we transformed the protocol slightly as a network protocol, see Algorithm 6 for the virtual source subprotocol.

Algorithm 6 Virtual source handling in η-Adaptive Diffusion.

```
Input: Previous virtual source $v_p$, Round identifier $r$, current
    step $s$, counter $h$
Environment: Neighbours $\mathcal{N}$ with $|\mathcal{N}| = \eta + 1$, Depth $d$
  for $v \in \mathcal{N} \setminus \{v_p\}$ do
    Send $m$ to $v$
    if $s + 1 \leq d$ and $s > 1$ then
      Send $m$ to $v$
    end if
  end for
  $h = h + 1$
  $s = s + 2$
  while $s \leq d$ do
    $s = s + 1$
    if $p(s, h) \leq \sim \mathcal{U}(0, 1)$ then
      for $v \in \mathcal{N}$ do
        Send $m$ to $v$
      end for
    else
      $v_{\text{next}} = \sim \mathcal{U}\{\mathcal{N} \setminus \{v_p\}\}$
      Send $(v_{\text{self}}, s, h, r)$ to $v_{\text{next}}$, to call Algorithm 6
      break
    end if
  end while
```

The virtual source token is forwarded with probability $p(s, h)$ based on the expected degree of the network, the current step s and amount of forwards h. The function to compute the probability is stated [21] as:

$$p(s, h) = \begin{cases} \frac{s - 2h + 2}{s + 2} & \text{if } \eta = 2, \\ \frac{(\eta - 1)^{\frac{s}{2} - h + 1} - 1}{(\eta - 1)^{\frac{s}{2} + 1} - 1)} & \text{else.} \end{cases}$$

To initiate the protocol, the initiating node v chooses a random neighbour. Then v sends the message m and the virtual source token transmission message $(v, s = 1, h = 1, r = \mathcal{H}(m))$, prompting the execution of Algorithm 6. Every virtual source should monitor the network for the progress of the protocol. A timeout will trigger retransmission to a different participant, as the previously selected might refuse cooperation or be unreachable. The timeout will extend on messages received through the protocol but only stop when receiving a flood and prune message.

C. Phase III: Flood and Prune Broadcast

The last virtual source initiates a flood and prune broadcast to ensure delivery to all nodes. When a node receives a message for the first time, it forwards the message to all neighbours, excluding the node which sent the message.

V. Security: Functionality under Attack

For the protocol to be considered secure and correct, all non-malicious nodes should receive a disseminated message. It is sufficient to show the last phase fulfils the requirement, and that the phase will be reached to show the desired property. Therefore, we work backwards from the last phase.

A. Attacker Model

The base model for the network, outlined in Section II, especially includes a connected graph after removing all malicious nodes from the network. Malicious nodes are interested in preventing a message from being broadcast. They are considered successful if some honest nodes do not receive the message, even though the network fulfils the requirements. Attackers are computationally limited, i.e., they are not able to break cryptographic primitives. As channels and transactions use strong cryptographic primitives, the restriction results in authenticated and secure channels and messages. Attackers act as participants of the network, not as internet service providers, hardware vendors or other outside entities. We do not consider vulnerabilities of implementations or general systems security.

B. Phase III: Flood and Prune Broadcast

When removing malicious nodes, the network remains a connected graph, based on the network requirements. Therefore, there exists a path between any node and the initiator of the flood and prune broadcast. The communication can be reduced to any two neighbouring nodes, with one receiving the message. The node will forward it to the neighbour, propagating along all available paths. This process is unaffected by non-participation or message injection. Therefore, we only require reaching the flood and prune stage and having an honest initiator to reach all nodes.

C. Phase II: Diffusion

We restrict the discussion to the virtual source sub-protocol. Reaching Phase 3 only hinges on this sub-protocol, so this restriction is warranted.

Let v_c be the current virtual source node and v_p the previous virtual source node. We can separate two cases. Either, v_c is fully uncooperative, i.e., sends no correct message. Alternatively, they are partly uncooperative by only sending correct messages back to v_p.

Fully uncooperative: v_p will not detect any correct messages. Therefore, the timeout of v_p will trigger, marking v_c as failed and continue the protocol itself. As a connected graph is available, even after removing all malicious or failed nodes, the protocol can continue eventually.

Partially uncooperative: v_p will not be able to distinguish this case from a fully functional run of the protocol. This situation is the case for all previous virtual source nodes. Either, the attacker switches to a flood and prune broadcast, but only forwarding it towards v_p or the attacker will send infinite diffusion messages. In the case of the flood and prune broadcast, the protocol is successful, due to the connected network. Further, v_p determines a maximum bound of messages they should observe, based on the state of h they received. A malicious node can therefore not send endless messages towards v_p to stall progress towards Phase 3.

Given this reselection, the last phase will be initiated if there is any available honest virtual source. Therefore, to reach all nodes, we only require reaching the diffusion stage and having an honest initiator of the diffusion stage.

D. Phase I: Dining Cryptographer Network

The first instance of the DCN corresponds to the protocol by von Ahn et al. [17] with $m = (r, \ell, K)$. Therefore, round one exhibits the same correctness, robustness, fairness and anonymity as the von Ahn protocol, which is secure in the discrete logarithm model. Whereby robustness means either the protocol succeeds, or an attacker is exposed, so the protocol will eventually succeed for finite attackers.

On successful transmission of the second instance, all participants will receive $[m_1, \dots, m_i]$. This can easily be verified:

$$
\begin{aligned}
Y_{result}[t] &\overset{Alg.\ 4}{=} \bigoplus_j S_j[t] \overset{Alg.\ 3}{=} \bigoplus_j \bigoplus_h s_{h,j}[t] \\
&= \bigoplus_h \bigoplus_j s_{h,j}[t] \overset{Alg.\ 5}{=} \bigoplus_h s_{h,k} \bigoplus_{j\in\{1\dots k-1\}} s_{k,j}[t] \\
&= \bigoplus_h (Y[t] \oplus \bigoplus_{j\in\{1\dots k-1\}} s_{k,j}[t]) \bigoplus_{j\in\{1\dots k-1\}} s_{k,j}[t] \\
&= \bigoplus_h Y[t] = [m_1, \dots, m_i].
\end{aligned}
$$

The commitments of round two are created using a PRNG seeded with K_i from the previous instance. If a collision occurs, the legitimate message sender can validate the commitments as zero commitments, as they know $r = K_i$ and the commitment. If a commitment does not reveal to be zero, the legitimate message sender will inject a blame message in the next protocol instance. The blame replaces the message identifiers r, ℓ, K and contains the seed, the blamed participant as well as a round identifier and message identifier. Other participants can verify the seed and that the commitment is not zero, and exclude the attacker. An honest participant can not be blamed without breaking the security assumption of the underlying commitments.

The case of non-cooperation by participants can be handled similar to von Ahn et al. [17] by sharing encrypted instances of all $\{(s_{\text{self},j}, r_{self,j})\}_j$ pairs with every participant, allowing for reconstruction of the contents with selective non-cooperation. This creates considerable additional load, so forming of a completely new group might be more economical.

As all sub-protocols are robust, we conclude 3P3 to either succeed or remove an attacker and therefore to be robust in its entirety.

E. Further Security Concerns

Besides the basic function of the protocol, we would like to note some security concerns. We do not consider confidentiality, as we are dealing with a broadcast protocol. Similarly, to keep messages smaller by default, we have no additional integrity checks, especially as these are often included in application protocols. Any application that requires integrity or confidentiality needs to provide application layer checks.

Lastly, a node holding the virtual source token can decide to start the last phase of the protocol early. Nodes, beside previous virtual source nodes, can not distinguish this situation from a valid phase switch, but the privacy impact and incentive for this attack are low.

VI. Privacy

For the sake of privacy discussion, we will use the notion of k-anonymity. Hereby k is the number of participants indistinguishable from the true originator, which optimally would be $k = |\text{participants}| - |\text{attackers}|$.

A. Independence of DCN and Diffusion

Further protocol steps should not provide further insights into the group to keep the privacy guarantees of the DCN. The second phase is started by transmission of the message m and the elements $v, s = 1, h = 1, r = \mathcal{H}(m)$. The elements $s = 1, h = 1$ identify that a new diffusion run is started but are independent of any contents, i.e., they are fixed for all possible runs. As v identifies the sender, v must be part of the originating DCN. Within the network, v was chosen by the originator, indirectly through a random identifier. This random identifier is not transmitted further, so no clues about group participants can be inferred. Therefore, the DCN can keep its privacy guarantees intact.

B. Global Passive Attacker

Revelations over recent years have shown that service providers and intelligence agencies across the world are collecting and analysing information, coming reasonably close to a global passive attacker. If such an observer could collect all slices $s_{i,j}$ of a participant, they could recompute the original message sent by the participant. This is prevented by authenticated encrypted channels between participants.

To prevent traffic correlation, DCNs requires all participants to send the same amount of data, not just the sender. While a global passive attacker can detect the communicating DC groups and the broadcast message, they can not identify the originator within the group [16], [17]. Against this attacker, phase one provides $k = |\text{group}|$-anonymity.

C. Dining Cryptographer Network Insider

To improve the detection of DC group participants, an attacker has to be part of the group. For β attackers within the group, DCN-based communication trivially provides $(k - \beta)$-anonymity. The anonymity guarantees depend on the group formation mechanism [17]. Current strategies of a random selection of participants with an assumed attacker probability p require group sizes of $\frac{2k}{1-p}$ for k-anonymity with high probability [17]. These bounds depend on the attacker probability distribution. External trust information during the group formation could improve the probability distribution.

D. Honest but Curious Botnet-like Attacker

The previous sections covered attack vectors against the DCN, i.e., an attacker on the outside and attackers on the inside. Therefore, we will focus on the privacy provided within the diffusion phase, as an improvement over the guarantees by the DCN.

An honest but curious attack could be performed by a single computer or a deployed botnet. The attackers connect to as many participants as possible and collect every message they observe. This behaviour can be seen often, e.g., by research groups [27] or information crawlers.

The original analysis of Fanti et al. [21] for adaptive diffusion gives a worst-case privacy estimate, as our modifications only limit the exposure to possible attackers. Based on this, we provide the argument for improved privacy over the DCN.

If an attacker were to receive the virtual source token, they could easily compute a candidate pool of d^h candidates, with h the current depth of the diffusion phase. For most sensible configurations, it holds for $h \geq 0$ that $d^h > \text{groupsize}$ improving the privacy of previous steps. Attackers with a penetration $p = \frac{|\text{attackers}|}{n}$ of the network, will receive approximately $p \times |\text{broadcasts}|$ of all first virtual source messages. This group increases further for participants of the spreading protocol, which never receive a virtual source message.

Consider two models: An attacker who creates a single connection to all nodes and distributed attackers with a network penetration of $p = \frac{|\text{attackers}|}{n}$ but normal behaviour. Let X be the random variable modelling the number of attackers chosen for a run of the spread sub-protocol. The probability $P(X \geq 1)$ of having at least one attacker included and the expected number of attackers per node $\mathbb{E}(X)$ are $\frac{\eta}{d}$ for a single attacker. For the distributed attack $P(X \geq 1) = 1 - (1-p)^\eta$ and $\mathbb{E}(X)\eta p$. At step t the expected amount of attackers hit are

$$\mathbb{E}_t(X) = \mathbb{E}(X) \sum_{i=1}^{t-1} \eta^i = \mathbb{E}(X)(\frac{\eta^t - 1}{\eta - 1} - 1).$$

This provides an estimation of the privacy impact of the parameter η.

VII. Performance

For the bandwidth usage during operation, we take a look at the message complexity, especially for the group phase. To assess the parameters and overall performance of our system, we used a simulator and parameter-dependent evaluations.

A. Group Size and Bandwidth

The most performance-critical part of our system is phase one. Due to the nature of DCNs, we expect the group phase to mostly scale with bandwidth with little influence by latency.

Given a commitment and randomness of 32 Bytes size, e.g., using Pedersen commitments [28], we can commit on 31 Bytes at a time. This results in $2 \times 2k(4 + 32k)$ bytes for the DC round and $2k \times 32$ bytes for commitments during round one. Round two requires $2 \times |m| \times (k-1)$ bytes for the DC round and $2 \times 32(k-1) \times \lceil |m| \frac{32}{31} \rceil$ bytes of commitments.

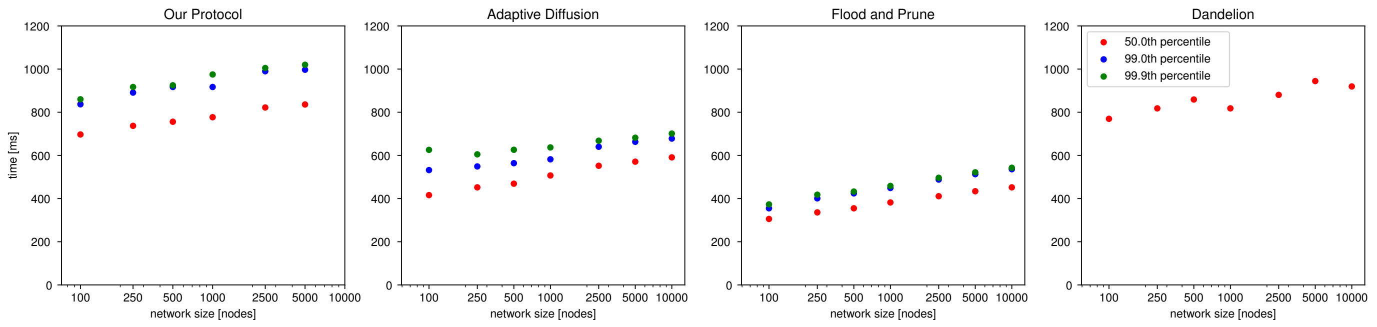

Fig. 2. The median, 99th percentile and 99.9th percentile of full protocol instance latency, i.e. until no more messages related to that protocol instance remain. Dandelion only shows the median, as the 99th percentile is far off the chart due to its random phase switch. Network sizes are scaled logarithmically.

Assuming a group size of 15, as used on the blockchain level in Monero, and a message length of 1024 Bytes, we determined the bandwidth consumption of phase one. A participant will send ≈ 1.34MiB of data and receive ≈ 18.83MiB for a full round. To only compute the first round of phase one, which is independent of any message size, a participant sends ≈ 434KiB and receives ≈ 6.09MiB. While these numbers can be handled quite easily by modern network devices, as they are on the scale of modern websites, it is easy to see that it does not scale well beyond sizes of around 30 to 40 participants. These sizes are sensible privacy settings, comparable to settings used in other strong systems [23].

B. *Simulation Methodology*

To evaluate our algorithm we built a discrete event simulation[1] for random networks. We implemented a skeleton of 3P3, where all message types and flows exist, but processing takes only simulated time, e.g., commitments take 0.5ms[2]. We implemented adaptive diffusion, flood and prune and a simplified Dandelion broadcast for comparison. Communication delay between nodes is based on a normal distribution $\mu = 80, \sigma = 15$ clamped between 20ms and 200ms, values estimated from well-connected clients of Hoiland et al. [29].

To generate a network, nodes create connections sequentially, until they have c connections. This construction results in c connections for most participants, but more for early participants.

We varied the parameters of the simulation by the number of participating nodes (100 to 10000), created connections between nodes (8 to 20), the depth of the diffusion phase (up to 8) and the spread of the diffusion (up to 8). We repeat all parameter combinations for 50 runs. A simulation run creates a random network and initiates a single protocol run until no messages are left.

C. *Network Size*

An overview of the results for scaling based on network size is shown in Figure 2, using the median, 99th and 99.9th percentile. This representation allows for analysis of the expected and worst-case performance of the protocols. Please note the logarithmic spacing of the x-axis.

The flood and prune entry provides a baseline for evaluation. The results show that the anonymity phases mostly dominate 3P3. Overall the performance of our system is reasonable, while noticeably slower than the baseline. We did not include results for the 99th percentile of Dandelion, which has, due to its random nature, fairly large outliers.

D. *Number of Connections and Diffusion Depth*

We chose the number of connections between nodes to stay above the $\log(n)$ connectedness boundary [19, Ch. 4]. The amount of connections is relevant to keep the preconditions intact as a fully connected network, even in the presence of malicious nodes. Minimum connections per node had a tiny impact on the overall performance.

Similar to the number of connections of nodes, the diffusion depth had little impact on overall results. The trend in runtime for longer diffusion runs was noticeably upwards, but negligible compared to other factors.

E. *Diffusion Spread*

We expected η to have some impact on the number of nodes reached over time, but not on total runtime. The visual analysis of Figure 3 indicates a difference in the base distribution for reaching 50% and 75% of the network. The results for 99% are inconclusive, which is expected. The means m_η and confidence intervals for $\eta = 2$ and $\eta = 8$ $m_2 = 640.23\text{ms} \pm 0.34\text{ms}$ and $m_8 = 615.60\text{ms} \pm 0.27\text{ms}$. support this interpretation, as they are strictly non-overlapping. This holds for all intermediate values of η as well. Overall the cost is small compared to the privacy gain of reducing η.

F. *Performance Optimizations*

In a sufficiently trustworthy environment, commitments and commitment seeds can be removed to reduce bandwidth consumption significantly. On repeated collisions, these protection mechanisms could be reintroduced.

Further, in an environment with at most $a < k - 1$ attackers and rare parallel transmissions, the number of message slots could be reduced to $2a + 2$ slots. This change would significantly reduce the required bandwidth As a cooperating

[1]Available here https://github.com/vs-uulm/netsim2.

[2]Average computation time of a commitment on our testing hardware.

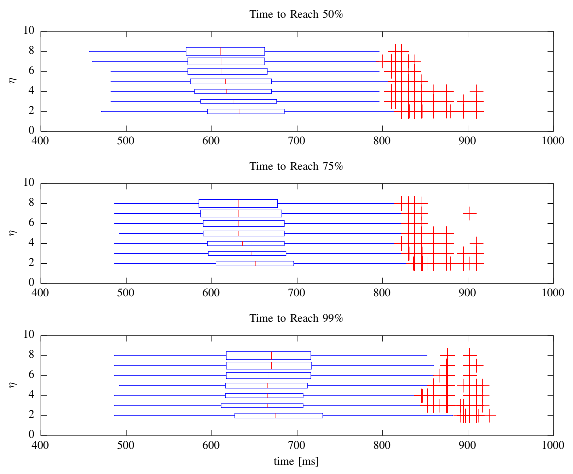

Fig. 3. Behaviour of the protocol based on η.

attackers could only block a transmission slots, leaving a success chance of $p > 0.5$.

VIII. Conclusion

In this paper, we proposed 3P3, a strong privacy protocol for broadcasting blockchain transactions. 3P3 can withstand a global passive observer and maliciously acting nodes jamming communication. The protocol can broadcast almost arbitrarily long messages with reduced overhead for zero messages of comparable proposals. We analysed the privacy and security of our system and provided a first performance analysis based on a simulation.

Our analysis shows comparable performance to Dandelion and only $\approx 2.5\times$ overhead over a flood and prune broadcast while disregarding bandwidth overhead. In usual environments of non-constant jamming attacks, this overhead can be significantly reduced. The performance impact analysis of the parameters of our system provides flexible parameters for system developers using 3P3 to customise the privacy-performance decisions.

While we focused on the use case of blockchains, 3P3 and our results apply to other domains as well. The system does not rely on blockchain specific properties, only on the privacy requirement of broadcast communication.

References

[1] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proc. of the Internet Meas. Conf.* ACM, 2013, pp. 127–140.

[2] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.

[3] S. Noether and S. Noether, "Monero is not that mysterious," Tech. Rep., 2014, https://lab.getmonero.org/pubs/MRL-0003.pdf.

[4] S. Noether, "Ring signature confidential transactions for monero," Crypt. ePrint Arch., Report 2015/1098, 2015.

[5] H. Kopp, D. Mödinger, F. J. Hauck, F. Kargl, and C. Bösch, "Design of a privacy-preserving decentralized file storage with financial incentives," in *Proc. of IEEE Sec. & Priv. on the Blockch. (S&B) (aff. with EUROCRYPT 2017)*. IEEE, 2017.

[6] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Proc. of the IEEE Symp. on Sec. and Priv. (SP)*. IEEE, 2013, pp. 397–411.

[7] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symp. on Sec. and Priv. (SP)*. IEEE, 2014, pp. 459–474.

[8] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Computer Security-ESORICS 2014*. Springer, 2014, pp. 345–364.

[9] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *Financial Cryptography and Data Sec.* Springer, 2014, pp. 486–504.

[10] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in zcash," in *27th USENIX Security Symposium*, Baltimore, MD, Aug. 2018, pp. 463–477.

[11] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using P2P network traffic," in *Proc. of the Int. Conf. on Finan. Cryp. and Data Sec.* Springer, 2014, pp. 469–485.

[12] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin p2p network," in *Proc. of the ACM Conf. on Comp. and Comm. Sec.* New York, NY, USA: ACM, 2014, pp. 15–29.

[13] S. Bojja Venkatakrishnan, G. Fanti, and P. Viswanath, "Dandelion: Redesigning the bitcoin network for anonymity," *Proc. of the ACM Measurement and Analysis of Comp. Sys. (POMACS)*, vol. 1, no. 1, pp. 22:1–22:34, Jun. 2017.

[14] B. Conrad and F. Shirazi, "A survey on tor and i2p," in *Ninth Int. Conf. on Internet Monitoring and Protection (ICIMP2014)*, 2014, pp. 22–28.

[15] D. Mödinger, H. Kopp, F. Kargl, and F. J. Hauck, "Towards enhanced network privacy for blockchains," in *Proc. of the 38th IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, 2018.

[16] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[17] L. von Ahn, A. Bortz, and N. J. Hopper, "K-anonymous message transmission," in *Proc. of the 10th ACM Conf. on Comp. and Comm. Sec.* New York, NY, USA: ACM, 2003, pp. 122–130.

[18] P. Golle and A. Juels, "Dining cryptographers revisited," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 456–473.

[19] M. O. Jackson, *Social and economic networks*. Princeton university press, 2010.

[20] G. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath, "Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees," *SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 1, pp. 5–7, Jan. 2019.

[21] G. Fanti, P. Kairouz, S. Oh, and P. Viswanath, "Spy vs. spy: Rumor source obfuscation," *SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 271–284, Jun. 2015.

[22] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable anonymous group messaging," in *Proc of the 17th ACM Conf. on Comp. and Comm. Sec.* New York, NY, USA: ACM, 2010, pp. 340–350.

[23] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale." in *OSDI*, 2012, pp. 179–182.

[24] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.

[25] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merritt, "Herd: A scalable, traffic analysis resistant anonymity network for voip systems," *Comp. Commun. Rev.*, vol. 45, no. 4, pp. 639–652, Aug. 2015.

[26] S. B. Mokhtar, G. Berthou, A. Diarra, V. Quéma, and A. Shoker, "Rac: A freerider-resilient, scalable, anonymous communication protocol," in *2013 IEEE 33rd Int. Conf. on Distributed Computing Systems*, July 2013, pp. 520–529.

[27] I. D. Mastan and S. Paul, "A new approach to deanonymization of unreachable bitcoin nodes," Crypt. ePrint Arch., Report 2018/243, 2018.

[28] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology — CRYPTO '91*, J. Feigenbaum, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 129–140.

[29] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring latency variation in the internet," in *Proc. of the 12th Int. Conf. on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 473–480.