Privacy in Population Protocols with Probabilistic Scheduling

Talley Amir[0009−0005−8404−7627] and James Aspnes[0000−0001−6188−1663]

Yale University, New Haven CT 06511, USA
{talley.amir,james.aspnes}@yale.edu

Abstract. The population protocol model [3] offers a theoretical framework for designing and analyzing distributed algorithms among limited-resource mobile agents. While the original population protocol model considers the concept of anonymity, the issue of privacy is not investigated thoroughly. However, there is a need for time- and space-efficient privacy-preserving techniques in the population protocol model if these algorithms are to be implemented in settings handling sensitive data, such as sensor networks, IoT devices, and drones. In this work, we introduce several formal definitions of privacy, ranging from assuring only plausible deniability of the population input vector to having a full information-theoretic guarantee that knowledge beyond an agent's input and output bear no influence on the probability of a particular input vector. We then apply these definitions to both existing and novel protocols. We show that the `Remainder`-computing protocol from [10] (which is proven to satisfy output independent privacy under adversarial scheduling) is not information-theoretically private under probabilistic scheduling. In contrast, we provide a new algorithm and demonstrate that it correctly and information-theoretically privately computes `Remainder` under probabilistic scheduling.

Keywords: Mobile ad-hoc networks · Population protocols · Information-theoretic privacy.

1 Introduction

Various issues arise when applying the theoretical population protocol model to real-world systems, one of the most critical of which is that of preserving privacy. The motivation for furthering the study of privacy within population protocols is to better adapt these algorithms to the real-world systems that they aim to model, such as sensor networks, systems of IoT devices, and swarms of drones, all of which handle sensitive data. Previous research in private population protocols only considers adversarial scheduling, which makes generous assumptions about our obliviousness to the scheduler's interaction choices and offers only very weak criteria for satisfying the definition of "privacy." In this work, we further refine these definitions considering a realistic range of threat models and security concerns under arbitrary schedules.

1.1 Related Work

Research in private computation within ad hoc networks is distributed (pun intended) over multiple academic fields. We limit our review of the literature to works that most closely relate to the theoretical model we study in this paper.

Population Protocols Privacy was introduced to the population protocol model in [10], where the authors define a notion of privacy called *output independent privacy* and provide protocols satisfying this definition for computing the semilinear predicates. Output independent privacy basically states that for any input vector and execution yielding a particular sequence of observations at an agent, there exists a different input vector and execution yielding the same sequence of observations at that agent. The practicality of this definition relies on adversarial scheduling, which allows the schedule of interactions to delay pairs of agents from interacting for an unbounded number of steps. Due to adversarial scheduling, the *existence* of an execution is sufficient to achieve plausible deniability: Agents have no metric for estimating time elapsed nor approximating how many interactions in which another agent has participated. Therefore, the observed state of an agent cannot be used to infer the agent's input as it may have deviated from its original state over the course of many interactions. However, if instead the scheduler is probabilistic, then there arises the issue of data leakage from inferring the population's interaction patterns.

Sensor Networks Population protocols are designed to model sensor networks, but there is a large body of literature on sensor networks that is not connected to the population protocol model. The capacity of agents in the domain of sensor networks is much larger than is assumed in population protocols; in particular, much of the privacy-preserving algorithms in this area involve encryption, which requires linear state space in the size of the population.

In recent years, viral exposure notification via Bluetooth has become a popular area of study [7, 9], and one that demands verifiable privacy guarantees due to widespread laws governing protected health data. However, the solutions in [7, 9] require centralization and high storage overhead. The closely related problem of anonymous source detection is studied in [5, 6]; however, these works require superconstant state space and only address this one task. Other research in wireless sensor networks investigates private data aggregation, which most closely resembles the goal of our research [8, 12, 15]. As before, these works require high computation and local memory as they implement their solutions using homomorphic encryption. Where alternative methods are used to avoid relying on encryption, a specialized network topology is needed for success [14] or only specific functions are computable [15].

While far from comprehensive, this sample of related works suggests that much of the research on privacy in wireless sensor networks is either limited by network topology or relies on computationally intensive encryption. For this reason, our goal is to develop privacy-preserving solutions for data aggregation in population protocols, bearing in mind the resource restrictions of the model.

1.2 Contribution

In this work, we study the privacy of population protocols in the random scheduling model. We demonstrate how existing privacy definitions fail under certain modelling assumptions, give new precise definitions of privacy in these settings, and offer a novel protocol in the uniform random scheduling population protocol model satisfying the new privacy definitions. In this work, we restrict our focus to computing the `Remainder` predicate. The proofs of all claims in this publication can be found in the extended version of our paper [1].

2 Preliminaries

A **population protocol** $\mathcal{P}$ is a tuple $(Q, \delta, \Sigma, \mathcal{I}, O, \mathcal{O})$ consisting of **state set** Q, **transition function** δ, **input set** Σ, **input function** $\mathcal{I}$, **output set** O, and **output function** $\mathcal{O}$ [3]. Protocols are run by a population, which consists of a set of n agents $\{A_j\}_{j=1}^n$ each with some input $i_j \in \Sigma$. At the start of the protocol, each agent converts its input to a state in Q via $\mathcal{I} : \Sigma \to Q$. In the early population protocol literature, $\mathcal{I}$ is only ever considered to be a deterministic function; however, in this work, we extend the model to allow for $\mathcal{I}$ to be randomized. The transition function $\delta : Q^2 \to Q^2$ designates how the agents update their states upon interacting with each other in pairs. As a shorthand for saying $\delta(q_1, q_2) = (q_1', q_2')$, we write $q_1, q_2 \to q_1', q_2'$ where δ is implied. The protocol aims to compute some function (whose output is in the output set O) on the initial inputs of the agents in the population. An agent's output value is a function of the agent's state, determined by $\mathcal{O} : Q \to O$.

The collection of agents' inputs is denoted as a vector $I \in \Sigma^n$, where each index of I reflects the input of a particular agent in the population. Adopting terminology from [10], we refer to I as an **input vector**. When the size of the state space is $O(1)$, the protocol cannot distinguish between two agents in the same state nor with the same input; therefore, we may want to refer to the multiset of input values in the input vector I, denoted multiset(I). After converting these inputs to elements of Q, the global state of the population is called a **configuration** and is represented as a vector $C \in Q^n$, where the i-th entry of the vector denotes the state of the i-th agent. Abusing notation, we say that $\mathcal{I}(I) = \langle \mathcal{I}(i_j) \rangle_{j=1}^n$ is the configuration resulting from applying the input function $\mathcal{I}$ to each of the agent inputs in $I = \langle i_j \rangle_{j=1}^n$.

Agents update their states via interactions with one another which are performed at discrete intervals, called **steps**. At each step, an ordered pair of agents (A_i, A_j) is selected from the population by the **scheduler**. To distinguish between the two agents in the ordered pair, we call the first agent the **Initiator** and the second the **Responder**. When an interaction takes place, the two selected agents update their states according to the transition function δ which may change the counts of states in the population, thereby updating the configuration. Let $\mathcal{C}$ be the configuration space, or the set of all possible configurations for a population of n agents with state space Q. We say that a configuration $D \in \mathcal{C}$ is **reachable** from $C \in \mathcal{C}$ via δ if there exists some series of ordered agent pairs such that starting from C, if the configuration is updated according to δ on those ordered pairs, then the resulting configuration is D [3]. If D is reachable from C, then we write $C \to D$. The infinite sequence of configurations resulting from the scheduler's infinite choice of interaction pairs is called an **execution**. An execution of a protocol is said to **converge** at a step τ when, for every step $t > \tau$, the output of each agent's state at t is the same as it is at τ (i.e. the output of every agent converges to some value and never changes thereafter). A stronger notion of termination is for a protocol to **stabilize**, meaning that after reaching some configuration C^*, the only configurations reachable from C^* result in the same outputs at every agent as in C^*. Abusing notation, we say $\mathcal{O}(C) = \lambda$ (or, the output of the *configuration* is λ) if $\mathcal{O}(q_j) = \lambda$ for every $q_j \in C$.

The goal of the population is to compute some function Φ on the input vector I, which means that the population eventually stabilizes towards a set of configurations $\mathcal{D} \subseteq \mathcal{C}$ for which $\mathcal{O}(D) = \Phi(I)$ for all $D \in \mathcal{D}$. The results of our work are commensurable with those of [10] which demonstrate that the semilinear predicates, which can be expressed using `Threshold` and `Remainder`, can be computed with output independent privacy under adversarial scheduling. Our work focuses on `Remainder`, defined for population protocols as follows:

Definition 1. *Given positive integers k and n, non-negative integer $r < k$, and input vector $I \in \mathbb{Z}_k^n$, let* `Remainder`$(I) = \text{True}$ *iff* $\sum_{j=1}^{n} i_j \equiv r \pmod{k}$.

The scheduler determines the pair of agents that interact at each step. The scheduler's choice of agent pairs may either be adversarial or probabilistic. An **adversarial scheduler** chooses pairs of agents to interact at each step as it desires, subject to a fairness condition. The condition used most commonly is called **strong global fairness**, and it states that if some configuration C occurs infinitely often, and $C \to C'$, then C' must occur infinitely often as well [3]. This means that if some configuration *can* occur, it eventually *must* occur, even if the adversarial scheduler wishes to delay its occurrence indefinitely. In works adopting adversarial scheduling, it can be claimed that a protocol eventually stabilizes to the correct answer, but not how quickly. A random or **probabilistic scheduler** instead selects pairs of agents to interact with one another according to some fixed probability distribution (usually uniform) over the ordered pairs of agents. Although population protocols consider interactions to occur in sequence, the systems they model typically consist of agents participating in interactions in parallel. As such, a natural estimation of **parallel time** is to divide the total number of interactions by n, as this roughly estimates the expected number of interactions initiated by a particular agent in the population.[1]

Our work crucially relies on distinguishing between an externally visible component of the agent state and a concealable secret state. Adopting notation from [2], we let S be the **internal state** space and M the set of **messages** which can be sent between the agents. Since each agent has both an internal and external state component, the total state space is then the Cartesian product of these sets $Q = S \times M$. This means that δ is instead a function computed lo-

[1] Under *non-uniform* random scheduling, this notion of time no longer applies.

cally at each agent according to its own state and the "message received" by its interacting partner $\delta : S \times M \times M \times \{\mathsf{Initiator}, \mathsf{Responder}\} \to S \times M$. This new mapping enforces the restriction that an agent can only use its received message to update its own state, and it does not observe the update to its interacting partner's state. For convenience, we use the original shorthand notation $\langle s_0, m_0\rangle, \langle s_1, m_1\rangle \to \langle s_0', m_0'\rangle, \langle s_1', m_1'\rangle$ to reflect the agents' state changes, where it is understood that the state update of A_b is computed independently of s_{1-b}.

3 Adversarial Model

In order to evaluate the extent to which private information can be learned by an observer in a population protocol, we must define the nature of the observer and its capabilities. In this work, we consider the agent inputs to be private information. We will consider the observer to take the form of an agent interacting in the protocol, meaning that it can observe the population only as other agents do, i.e., by participating in interactions as they are slated by the scheduler. However, we do not preclude the possibility that the observer may have greater computational capabilities than ordinary honest agents, and may therefore infer additional information from its observed history of interactions. We assume that the observer is **semi-honest**, meaning that it must adhere to the protocol rules exactly, but may try to infer additional knowledge from the system [11]. As such, the observer can only gather knowledge by interacting with other agents as prescribed by the transition function δ.

Since an observer presents as an agent in the population, we can imagine that multiple adversaries may infiltrate the system. However, we restrict that each observer be **non-colluding**, meaning that it cannot communicate with other nodes in the network besides participating in the protocol interactions honestly. This is because otherwise we could imagine that an observer may disguise itself as multiple agents in the population making up any fraction of the system. Although not studied within this work, it is of interest to find bounds on the fraction of agents that can be simulated by the observer in any network and still successfully hide honest agents' inputs. Notice that the restriction that the observer is both semi-honest and non-colluding is equivalent to assuming that there is only one such agent in the population, because from the point of view of the observer, all other agents appear to be honest.

Finally, we allow a distinction between externally visible messages and internally hidden states as in [2] to allow agents to conceal a portion of their states toward the end goal of achieving privacy. The distinction between messages and the internal state will be crucial to studying privacy in the population model as without it, there is no mechanism for hiding information from an observer.

4 Definitions of Input Privacy

In this section, we examine definitions of privacy in population protocols under adversarial and probabilistic scheduling given our specified adversarial model.

4.1 Output Independent Privacy

The privacy-preserving population protocol from [10] operates under the adversarial scheduling model and uses constant state-space. Therefore, [10] demonstrates privacy in the context of computing semilinear predicates only. The authors offer a formal definition of input privacy under these circumstances called **output independent privacy**, defined as follows:

> "A population protocol has this property if and only if there is a constant n_0 such that for any agent p and any inputs I_1 and I_2 of size at least n_0 in which p has the same input, and any execution E_1 on input I_1, and any T, there exists an execution E_2 on input I_2, such that the histories of p's interactions up to T are identical in E_1 and E_2."

Essentially, this definition states that a semi-honest process p cannot tell whether the input vector is I_1 or I_2 given its sequence of observations because either input could have yielded the same observations under an adversarial scheduler.

Output independent privacy is a successful measure in [10] because the scheduling in that work is assumed to be adversarial, therefore no inference can be made about the interaction pattern. The authors leverage this to achieve privacy which is best framed as "plausible deniability" – an agent may directly observe another agent's input, but the unpredictability of the scheduler disallows the observer to claim with certainty that the observed value is indeed the input.

This argument breaks down when the scheduler is probabilistic because now an agent can infer a probability distribution on the interaction pattern, and thus also infer a probability distribution on the input value of the agent's interacting partner. In light of this insight, we now introduce novel definitions for the purpose of assessing privacy in population protocols with probabilistic scheduling.

4.2 Definitions of Privacy Under Probabilistic Schedules

Consider an agent $\mathcal{A}$ with initial state $q_0^{\mathcal{A}} = (s_0^{\mathcal{A}}, m_0^{\mathcal{A}})$. Given its sequence of observed messages and the role ($\mathsf{Initiator}$ or $\mathsf{Responder}$) played by $\mathcal{A}$ in each interaction, $\mathcal{A}$ can deterministically compute each of its subsequent state updates. Let's call these messages (observed by $\mathcal{A}$) $o_1^{\mathcal{A}}, o_2^{\mathcal{A}}, o_3^{\mathcal{A}}, ...$, and denote by $q_\varepsilon^{\mathcal{A}} = \delta(\rho_\varepsilon^{\mathcal{A}}, s_{\varepsilon-1}^{\mathcal{A}}, m_{\varepsilon-1}^{\mathcal{A}}, o_\varepsilon^{\mathcal{A}}) = (s_\varepsilon^{\mathcal{A}}, m_\varepsilon^{\mathcal{A}})$ the updated state of $\mathcal{A}$, originally in state $q_{\varepsilon-1}^{\mathcal{A}} = (s_{\varepsilon-1}^{\mathcal{A}}, m_{\varepsilon-1}^{\mathcal{A}})$, upon interacting as $\rho_\varepsilon^{\mathcal{A}} \in \{\mathsf{Initiator}, \mathsf{Responder}\}$ with another agent with message $o_\varepsilon^{\mathcal{A}}$ in its ε-th interaction. Adopting notation from [11], we denote the **view** of an agent $\mathcal{A}$ participating in protocol $\mathcal{P}$ in an execution E by $\mathsf{view}_{\mathcal{A}}^{\mathcal{P}}(E) = \langle i_{\mathcal{A}}; q_0^{\mathcal{A}}; (\rho_1^{\mathcal{A}}, o_1^{\mathcal{A}}), (\rho_2^{\mathcal{A}}, o_2^{\mathcal{A}}), ...\rangle$. This view consists of $\mathcal{A}$'s input, the initial state of $\mathcal{A}$, and a list of $\mathcal{A}$'s interactions over the course of the execution, from which every subsequent state of $\mathcal{A}$ can be computed.[2]

Let $\boldsymbol{\mathsf{view}}_{\mathcal{A}}^{\mathcal{P}}(\boldsymbol{C})$ be a random variable representing the view of agent $\mathcal{A}$ drawn uniformly from all realizable executions starting from configuration C resulting

[2] For randomized δ, we assume $\mathcal{A}$ has a fixed tape of random bits that it uses to update its state, so $\mathcal{A}$ can still reconstruct its entire view from the specified information.

from the possible randomness used by the scheduler. Similarly, let $\boldsymbol{view}_{\mathcal{A}}^{\mathcal{P}}(\boldsymbol{I})$ be a random variable representing the view of agent $\mathcal{A}$ drawn from all possible executions starting from any configuration C in the range of $\mathcal{I}(I)$ according to the probability distribution given by the randomness of $\mathcal{I}$. In general, we use the convention that random variables appear in mathematical boldface.

Privacy, like many other security-related key terms, has a wide range of technical interpretations. As such, we now offer several distinct formal definitions of privacy in the population model.

Plausible Deniability Perhaps the weakest form of privacy we can possibly define is that of *plausible deniability*, meaning that an adversary always doubts its guess of an agent's input value (even if it has unbounded resources). This is not a novel concept [10, 13], but in the context of input vector privacy for probabilistic population protocols, we define this notion as follows:

Let $\mathcal{M}_\lambda = \{\text{multiset}(I) : \Phi(I) = \lambda\}$ be the set of all distinct multisets of inputs whose corresponding input vector evaluates to λ,[3] and let $\mathcal{M}_\lambda^\kappa = \{\text{multiset}(I) : \text{multiset}(I) \in \mathcal{M}_\lambda \wedge \kappa \in \text{multiset}(I)\}$ be the set of all distinct multisets of inputs outputting λ which contain at least one input equal to κ.

Definition 2. *Let $\mathcal{P}$ be a population protocol on n agents with input set Σ and let $\mathcal{D}$ be any probability distribution on input vectors in Σ^n. Then $\mathcal{P}$ is **weakly private** if for every distribution $\mathcal{D}$ on Σ^n, every non-colluding semi-honest unbounded agent $\mathcal{A}$ in a population of size n executing $\mathcal{P}$, and for any view $V = \langle i; q; \{(\rho_\varepsilon^{\mathcal{A}}, o_\varepsilon^{\mathcal{A}})\}\rangle$ with output λ (as determined from the view V) and with $|\mathcal{M}_\lambda^i| > 1$, there exist I_1 and I_2 in $\mathcal{S}_\lambda$ such that*

1. *both* $\text{multiset}(I_1)$ *and* $\text{multiset}(I_2)$ *are elements of* $\mathcal{M}_\lambda^i$,
2. $\text{multiset}(I_1) \neq \text{multiset}(I_2)$, *and*
3. $\Pr(\boldsymbol{view}_{\mathcal{A}}^{\mathcal{P}}(\boldsymbol{I_1}) = V) = \Pr(\boldsymbol{view}_{\mathcal{A}}^{\mathcal{P}}(\boldsymbol{I_2}) = V)$,

where the probabilities in the final condition are taken over $\mathcal{D}$, the randomness of $\mathcal{I}$, and the uniform randomness of the scheduler.

In plain English, Definition 2 says that any agent participating in the protocol cannot simply guess the "most likely" input vector because for each such vector, pending certain circumstances, there exists a distinct input vector yielding the same views for that agent with the same probabilities. This definition differs from output independent privacy [10] in that it considers adversarial strategies for guessing the input vector which rely on distributional data collected from interactions with other agents.

The condition $|\mathcal{M}_\lambda^i| > 1$ necessitates that weak privacy may only hold for multisets of inputs for which plausible deniability is even possible. For example, if the output of the computation for the `Or` predicate is 0, then there is only one possible multiset of inputs that could have yielded this outcome, so there is no denying what the input vector must have been (namely, the all-zero vector).

[3] Recall that agents in the same state are indistinguishable by the protocol; therefore, Φ must map any input vectors with the same multiset of inputs to the same output.

Information-Theoretic Input Privacy A stronger notion of privacy is one that claims that an observer cannot narrow down the possibility of input vectors at all based on its observations. This prompts our next definition.

Let $\mathcal{P}$ be a population protocol with input set Σ and let $\mathcal{D}$ be a probability distribution on input vectors in Σ^n. Let $\boldsymbol{I} \sim \mathcal{D}$ be a random variable representing the selected input vector. Additionally, let $\boldsymbol{i_{\mathcal{A}}}$ and $\boldsymbol{\lambda_{\mathcal{A}}}$ be random variables representing the input and output at agent $\mathcal{A}$, and let $\boldsymbol{view}_{\mathcal{A}}^{\mathcal{P}}(\boldsymbol{i}, \boldsymbol{\lambda})$ be a random variable representing the view of agent $\mathcal{A}$ participating in an honest execution of $\mathcal{P}$ that is consistent with a fixed input i at $\mathcal{A}$ and observed output λ.

Definition 3. *Protocol $\mathcal{P}$ satisfies **information-theoretic input privacy** if for every non-colluding semi-honest unbounded agent $\mathcal{A}$ and every input $i \in \Sigma$, output $\lambda \in O$, view V, input vector $I \in \mathcal{S}_\lambda$, and distribution $\mathcal{D}$ on Σ^n,*

$$Pr(\boldsymbol{I} = I \mid \boldsymbol{view}_{\mathcal{A}}^{\mathcal{P}}(\boldsymbol{i}, \boldsymbol{\lambda}) = V) = Pr(\boldsymbol{I} = I \mid \boldsymbol{i_{\mathcal{A}}} = i, \boldsymbol{\lambda_{\mathcal{A}}} = \lambda),$$

where V is consistent with input i and output λ.

The above definition essentially states that conditioned on knowing one's own input and the output of the computation, the rest of the agent's view in the protocol's computation gives no advantage in guessing the input vector.

We offer another definition of privacy called **input indistinguishability** in the extended version of this paper that is independent of our main results.

Intuitively, it is straightforward to see that information-theoretic privacy is the strongest of the definitions discussed in this section (proof in full paper):

Theorem 1. *If $\mathcal{P}$ is information-theoretically private, then $\mathcal{P}$ also satisfies output independent privacy, weak privacy, and input indistinguishability.*

5 Private `Remainder` with Adversarial Scheduling

As a means for comparison, we analyze the `Remainder` protocol from [10], shown in Algorithm 1. The protocol does not distinguish between internal state space and message space, so the entirety of each agent's state is seen by its interacting partner. The agent states are tuples (v, f), initially $(i_j, 1)$, where v is the value of the agent and f is a flag bit denoting whether or not the agent has decided its output yet. The protocol accumulates the total sum (modulo k) of all agents' inputs by transferring values in units rather than in full in a single interaction. As shown in (M1), the protocol subtracts 1 (modulo k) from one of the inputs and adds it to the other input, maintaining the invariant that the sum of all the values in the population is the same at each step. Because all computations are done modulo k, (M1) can be repeated indefinitely. Transitions (M2) and (M3) handle the flag bit (where $*$ is a wildcard that can match any value), ensuring that (M1) occurs an unbounded but finite number of times. The output values are $\{\bot_0, \bot_1\}$, denoting that the predicate is FALSE or TRUE, respectively. The protocol converges when all but one agent has $\bot_0$ or $\bot_1$ as their value.

$$(v_1, 1), (v_2, 1) \to (v_1 + 1, 1), (v_2 - 1, 1) \qquad \text{(M1)}$$
$$(*, 1), (*, *) \to (*, 0), (*, *) \qquad \text{(M2)}$$
$$(*, 0), (*, 1) \to (*, 1), (*, 1) \qquad \text{(M3)}$$
$$(v_1, 0), (v_2, 0) \to (v_1 + v_2, 0), (0, 0) \qquad \text{(M4)}$$
$$(v_1, 0), (0, 0) \to (v_1, 0), (\perp_0, 0) \qquad \text{(M5)}$$
$$(\perp_i, *), (*, 1) \to (0, 0), (*, 1) \qquad \text{(M6)}$$
$$(r, 0), (\perp_i, 0) \to (r, 0), (\perp_1, 0) \qquad \text{(M7)}$$
$$(v_1, 0), (\perp_i, 0) \to (v_1, 0), (\perp_0, 0), \text{ if } v_1 \neq r \qquad \text{(M8)}$$

Algorithm 1: Output Independent Private `Remainder` [10]

The crux of the proof that Algorithm 1 satisfies output independent privacy focuses on transition (M1). When an adversarial process p interacts with an honest agent A in state (v, f), p cannot know how close v is to A's original input because, for $n \geq 3$, we can construct multiple executions wherein A has value v upon interacting with p. For example, we can construct an execution where some agent B transfers as many units to A via (M1) as needed to get A's value to be v, and as long as p and B do not interact with each other before p interacts with A, p's view is the same in this execution.

However, output independent privacy does not successfully carry over to the random scheduling model because we can no longer construct *any* execution "fooling" the process p, as some such executions are of very low probability. For instance, the probability that agents A and B interact v' times in a row, during which time p does not interact with B at all, becomes small for large values of v'. This means that it is less probable that an agent's value will deviate from its original input value early on in the execution.

6 Private `Remainder` with Probabilistic Scheduling

In this section, we introduce a novel algorithm for information-theoretically privately computing `Remainder` in the population protocol model with probabilistic scheduling. Our algorithm is inspired by the famous example of cryptographically secure multiparty computation of `Remainder` in a ring network. We refer to this algorithm as RingRemainder, and it works as follows:

There are n agents $A_1, ..., A_n$ arranged in a circle. Agent A_1 performs the leader's role, which is to add a uniformly random element $r \in \mathbb{Z}_k$ to their input and pass the sum (modulo k) to agent A_2. For each remaining agent A_i, upon receiving a value from A_{i-1}, A_i adds its own input to that value and passes the resulting sum to $A_{i+1 \pmod n}$. When A_1 receives a value from A_n, it subtracts r and broadcasts the result to everyone. Suppose the agents have inputs $i_1, ..., i_n$.

Then A_1 sends $m_1 = i_1 + r$ to A_2, A_2 sends $m_2 = i_1 + r + i_2$ to A_3, and so on, until A_n sends $m_n = r + \sum_{j=1}^{n} i_j$ to A_1. Thus, the value broadcast to all agents $m_n - r$ is exactly equal to $\sum_{j=1}^{n} i_j$, the sum of the agents' inputs modulo k. Assuming honest participants and secure pairwise communication, this protocol achieves information-theoretic input privacy (see extended paper for proof).

We now adapt this scheme to compute `Remainder` in the population model with information-theoretic privacy.

Algorithm Overview Our protocol simulates the transfer of information exactly as in RingRemainder. We assume that the protocol has an initial leader with a special token that circulates the population. Each time an agent receives the token and some accompanying value, it adds its input to that value and passes the sum, along with the token, to another agent. This means the current owner of the token holds the aggregate sum of the agents' inputs who previously held the token. When an agent passes the token to another agent, it labels itself as "visited" so as to ensure that its input is included in the sum exactly one time. Once the token has visited all of the agents, it is returned to the leader (along with the total sum of all of the agents' inputs). In order to achieve this functionality, there are two crucial obstacles we must overcome:

First, we need a mechanism for securely transferring a message between two agents such that no other agent learns the message except the sender and the intended recipient. This task is nontrivial because population protocols do not allow agents to verify a condition before transmitting a message in an interaction; it is assumed that the message exchange and state update occur instantaneously. To do this, we provide a secure peer-to-peer transfer subroutine in Section 6.1.

Second, we need a way to determine whether or not every agent in the population has been visited by the token. When this happens, we want the final token owner to pass the token back to the leader so that the leader can remove the randomness it initially added to the aggregate that has been passed among the agents. We must try to prevent passing the aggregate back to the leader before all inputs have been incorporated into the aggregate as this would cause some agents to be excluded from the computation. In order to achieve this, we use the probing protocol from [4] which we describe in further detail in Section 6.2.

Leveraging these two subroutines, we design our main algorithm for computing `Remainder` with information-theoretic privacy in Section 6.3.

6.1 Secure Peer-to-Peer Transfer

In order for our algorithm to guarantee input privacy, the communication of the intermediate sums between any two agents must remain secure. Here we introduce a novel secure peer-to-peer transfer protocol, defined as follows:

Definition 4. *Let M be a message space, $\mathcal{D}$ be some distribution on M, and I be any fixed input vector in Σ^n. A* ***secure peer-to-peer transfer routine***

$$\langle \mu, (r, \mathfrak{S})\rangle, \langle *, (*, \overline{\mathfrak{u}})\rangle \to \langle \mu, (r', \mathfrak{S})\rangle, \langle *, (*, \overline{\mathfrak{u}})\rangle \tag{S1}$$
$$\langle \mu, (r, \mathfrak{S})\rangle, \langle *, (*, \mathfrak{u})\rangle \to \langle \perp, (\mu - r, \mathfrak{S}')\rangle, \langle r, (*, \mathfrak{R})\rangle \tag{S2}$$
$$\langle \perp, (x, \mathfrak{S}')\rangle, \langle y, (*, \mathfrak{R})\rangle \to \langle \perp, (\perp, \overline{\mathfrak{u}})\rangle, \langle x + y, (*, \mathfrak{S})\rangle \tag{S3}$$

Algorithm 2: Population Protocol for Secure P2P Transfer

is a protocol $\mathcal{P}$ that transfers data $m \xleftarrow{\mathcal{D}} M$ from one agent Sender *to another* Receiver *such that there exist PPT algorithms W_1, W_2 where*

$$\Pr\left(W_1(\textbf{\textit{view}}^{\mathcal{P}}_{\textsf{\textit{Sender}}}(\boldsymbol{I})) = m\right) = \Pr\left(W_2(\textbf{\textit{view}}^{\mathcal{P}}_{\textsf{\textit{Receiver}}}(\boldsymbol{I})) = m\right) = 1$$

and for all $i : A_i \notin \{$Sender, Receiver$\}$ and PPT algorithm W'

$$\Pr\left(W'(\textbf{\textit{view}}^{\mathcal{P}}_{\boldsymbol{A_i}}(\boldsymbol{I})) = m\right) = \Pr(m \xleftarrow{\mathcal{D}} M)$$

In other words, a secure peer-to-peer transfer routine allows a Sender to transfer a message m to a Receiver such that only Sender and Receiver are privy to m and all other agents cannot guess m with any advantage over knowing only the *a priori* distribution on the message space.

Our Algorithm 2 satisfies this definition: Each agent's state $\langle \mu, (r, L)\rangle$ consists of a hidden secret μ, and a public randomness value r and label L. The goal of the protocol is to pass a secret message from one agent (marked as Sender with label $\mathfrak{S}$, of which there may only be one in the population) to another agent meeting some specified criteria labeled by $\mathfrak{u}$, of which there may be any number (including zero). Until the Sender meets an agent with label $\mathfrak{u}$, it refreshes its randomness at each interaction to ensure that the randomness it transmits to the Receiver is uniform (S1). When the Sender finally meets some agent with $\mathfrak{u}$, it marks that agent as the Receiver and transmits r; it also updates its own token to $\mathfrak{S}'$ to remember that it has met and labeled a Receiver (S2). Then, the Sender waits to meet the Receiver again, at which point it gives it a message masked with the randomness it sent in the previous interaction and marks itself with the label $\overline{\mathfrak{u}}$ to signify the end of the transmission (S3). By the end of the protocol, exactly one agent is selected as the Receiver and stores μ internally. The protocol has state space $(\mathbb{Z}_k \cup \{\perp\})^2 \times \{\mathfrak{S}, \mathfrak{S}', \mathfrak{R}, \mathfrak{u}, \overline{\mathfrak{u}}\}$, which for constant k is of size $O(1)$. As such, we conclude (and prove in the extended paper):

Theorem 2. *Algorithm 2 is a secure peer-to-peer transfer routine.*

6.2 Probing Protocol

In order to adapt RingRemainder to the population protocol model, we need a way to detect when every agent has been included in the aggregation so the final sum can be passed back to the leader. To do this, we use a probe.

A **probing protocol**, or **probe**, is a population protocol that detects the existence of an agent in the population satisfying a given predicate [4]. In essence, the probe (initiated by the leader) sends out a 1-signal through a population of agents in state 0. If the 1-signal reaches an agent satisfying the predicate, that agent initiates a 2-signal which spreads back to the leader by epidemic. Higher number epidemics overwrite lower ones, so if some agent in the population satisfies π then the leader eventually sees the 2-signal. The probe, used in conjunction with the phase clock from the same work [4], allows the leader to detect the presence of an agent satisfying π in $O(n \log n)$ interactions using $O(1)$ states with probability $1 - n^{-c}$ for any fixed constant $c > 0$.

We define the "output" of the protocol (computed only at the leader) to be 0 for states 0 and 1, and 1 for state 2 (i.e. the leader's probe outputs 1 if and only if some agent in the population satisfies π). At the start of each round of the phase clock, agents reset their value to 0 and the leader initiates a new probe. Both the probe and the phase clock states are components of the message space, and the transitions for these subroutines are independent of the transitions for the main protocol, so we consider the two "protocols" to be taking place in parallel.

6.3 `Remainder` with Information-Theoretic Privacy

We provide here a novel algorithm which computes `Remainder` and achieves *information-theoretic input privacy* in the population protocol model with high probability, assuming a uniform random scheduler.

First, each agent applies the input function $\mathcal{I}$ to their input as follows:

$$\mathcal{I}(i_j, \ell) = \begin{cases} \langle i_j + r^0, (r^j, \mathfrak{S}, 1, Z = Z_0)\rangle & \ell = 1 \\ \langle i_j, (r^j, \mathfrak{u}, 0, Z = Z_0)\rangle & \ell = 0 \end{cases}$$

where r^j is drawn uniformly at random from $\mathbb{Z}_k$ for $j \in \{0, 1, ..., n\}$, and Z (initialized to Z_0) is a probe subroutine (including its associate phase clock). The input function assumes an initial leader, specified by $\ell = 1$. The components of the state $\langle \mu, (r, L, \ell, Z)\rangle$ are μ (the hidden internal component of the state called the **secret**), r (the **mask**), L (the agent's **label**), ℓ (the **leader bit**), and Z (the **probe**). The transitions describing the protocol can be found in Algorithm 3.

The general structure of the transitions from the secure peer-to-peer transfer protocol in Algorithm 2 is used to send the intermediate sums in (R1), (R2), and (R3). However, instead of just storing the message received, the Receiver computes the sum of the message and its own input and stores the result internally. Each subsequent Sender searches the population for an agent whose input has not yet been incorporated into the sum (signified by the $\mathfrak{u}$ state). When no one in the population has $\mathfrak{u}$ anymore, the probe detects this and outputs 1 at the leader from this point onward.

Although not shown, each interaction also performs an update to the probing subroutine by advancing the phase clock and probe at both agents in the interaction. When the probe begins to output 1, with high probability every agents' label is set to $\overline{\mathfrak{u}}$, alerting the leader to set its label to $\mathfrak{u}$. This makes the leader the

$$\langle *, (r, \mathfrak{S}, *, *)\rangle, \langle *, (*, \bar{\mathfrak{u}}, *, *)\rangle \to \langle *, (r', \mathfrak{S}, *, *)\rangle, \langle *, (*, \bar{\mathfrak{u}}, *, *)\rangle \tag{R1}$$

$$\langle u, (r, \mathfrak{S}, *, *)\rangle, \langle v, (*, \mathfrak{u}, *, *)\rangle \to \langle \bot, (u - r, \mathfrak{S}', *, *)\rangle, \langle v + r, (*, \mathfrak{R}, *, *)\rangle \tag{R2}$$

$$\langle *, (x, \mathfrak{S}', *, *)\rangle, \langle y, (*, \mathfrak{R}, *, *)\rangle \to \langle *, (\bot, \bar{\mathfrak{u}}, *, *)\rangle, \langle x + y, (*, \mathfrak{S}, *, *)\rangle \tag{R3}$$

$$\langle \bot, (\bot, \bar{\mathfrak{u}}, 1, 1)\rangle, \langle *, (*, *, *, *)\rangle \to \langle \bot, (\bot, \mathfrak{u}, 1, 1)\rangle, \langle *, (*, *, *, *)\rangle \tag{R4}$$

Algorithm 3: Information-Theoretically Private `Remainder`

only agent able to be the next Receiver. When the leader receives the final value stored at the Sender, the leader can place the answer into a separate portion of the external state (not shown in Algorithm 3) so that all other agents can copy it, which takes $O(n^2 \log n)$ additional steps with high probability. The leader must also have an additional component to its *hidden* state which stores the randomness used in its initial message transfer (also not shown in Algorithm 3).

The correctness and privacy guarantees of Algorithm 3 are stated below (see extended paper for proofs):

Theorem 3. *For any fixed $c > 0$, Algorithm 3 computes* `Remainder` *in a population of size n in $\Theta(n^3 \log n)$ steps with probability at least $1 - n^{-c}$.*

Theorem 4. *When Algorithm 3 correctly computes the* `Remainder` *predicate, it satisfies information-theoretic input privacy.*

If the protocol fails due to a phase clock error in the probing subroutine, we actually do not know how much information is leaked by the protocol, though we suspect it to be limited. We designate this as outside of the scope of this work and only make claims about privacy when the protocol succeeds. Note that it is impossible to achieve information-theoretic privacy with probability 1 in asynchronous distributed systems because there is always the possibility of premature termination due to indefinite exclusion of agents from the protocol.

7 Conclusion

In this work, we offer various new security definitions in population protocols, such as multiple definitions of privacy which accommodate a range of threat models and scheduling assumptions, and a formal definition of secure peer-to-peer communication. We also develop algorithms solving secure pairwise communication in the model and information-theoretically private computation of the `Remainder` predicate. In order to show that we can achieve information-theoretic privacy (with high probability) for all semilinear predicates, as in [10], similar algorithms for computing `Threshold` and `Or` are also needed. We leave these problems as open for future work.

Bibliography

[1] Amir, T., Aspnes, J.: Privacy in population protocols with probabilistic scheduling (2023), https://arxiv.org/abs/2305.02377

[2] Amir, T., Aspnes, J., Doty, D., Eftekhari, M., Severson, E.: Message Complexity of Population Protocols. In: 34th International Symposium on Distributed Computing (DISC 2020). Leibniz International Proceedings in Informatics (LIPIcs), vol. 179, pp. 6:1–6:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.DISC.2020.6

[3] Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Proceedings of the Annual ACM Symposium on Principles of Distributed Computing **18**, 235–253 (03 2006). https://doi.org/10.1007/s00446-005-0138-3

[4] Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distributed Computing **21**, 183–199 (01 2006). https://doi.org/10.1007/s00446-008-0067-z

[5] Aspnes, J., Diamadi, Z., Gjøsteen, K., Peralta, R., Yampolskiy, A.: Spreading alerts quietly and the subgroup escape problem. In: Advances in Cryptology - ASIACRYPT 2005. pp. 253–272. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[6] Blazy, O., Chevalier, C.: Spreading alerts quietly: New insights from theory and practice. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. ARES 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3230833.3230841

[7] Canetti, R., Kalai, Y.T., Lysyanskaya, A., Rivest, R., Shamir, A., Shen, E., Trachtenberg, A., Varia, M., Weitzner, D.J.: Privacy-preserving automated exposure notification. IACR Cryptol. ePrint Arch. **2020**, 863 (2020)

[8] Castelluccia, C., Mykletun, E., Tsudik, G.: Efficient aggregation of encrypted data in wireless sensor networks. In: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. pp. 109–117 (2005)

[9] Chan, J., Foster, D., Gollakota, S., Horvitz, E., Jaeger, J., Kakade, S., Kohno, T., Langford, J., Larson, J., Sharma, P., Singanamalla, S., Sunshine, J., Tessaro, S.: Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing (2020)

[10] Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Ruppert, E.: Secretive birds: Privacy in population protocols. In: Principles of Distributed Systems. pp. 329–342. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[11] Lindell, Y.: How to Simulate It: A Tutorial on the Simulation Proof Technique, pp. 277–346. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-57048-8_6

[12] Liu, C.X., Liu, Y., Zhang, Z.J., Cheng, Z.Y.: High energy-efficient and privacy-preserving secure data aggregation for wireless sensor networks. International Journal of Communication Systems **26**(3), 380–394 (2013). `https://doi.org/10.1002/dac.2412`

[13] Monshizadeh, N., Tabuada, P.: Plausible deniability as a notion of privacy. In: 2019 IEEE 58th Conference on Decision and Control (CDC). pp. 1710–1715 (2019). `https://doi.org/10.1109/CDC40024.2019.9030201`

[14] Setia, P.K., Tillem, G., Erkin, Z.: Private data aggregation in decentralized networks. In: 2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG). pp. 76–80 (2019). `https://doi.org/10.1109/SGCF.2019.8782377`

[15] Taban, G., Gligor, V.D.: Privacy-preserving integrity-assured data aggregation in sensor networks. In: Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 03. p. 168–175. CSE '09, IEEE Computer Society, USA (2009). `https://doi.org/10.1109/CSE.2009.389`