Bazaar: Anonymous Resource Sharing

Christoph Coijanovic
Daniel Schadt
firstname.lastname@kit.edu
Karlsruhe Institute of Technology
Germany

Christiane Weis
firstname.lastname@neclab.eu
NEC Laboratories Europe
Germany

Thorsten Strufe
firstname.lastname@kit.edu
Karlsruhe Institute of Technology
Germany

ABSTRACT

In areas such as manufacturing or logistics, it is beneficial for everyone to share access capacity with others. Increased efficiency increases profits, lowers prices for consumers, and reduces environmental impact. However, in order to share a resource such as manufacturing capacity, suitable partners must be found. Ideally, a centralized exchange is used to find partners, but this comes with privacy risks. Since participants in the exchange are competitors, they can use information about someone else's capacity to their disadvantage, e.g., by undercutting the prices of an already poorly performing competitor to drive it out of business. In this paper, we show that such an exchange can be set up without compromising the privacy of its participants. We formalize privacy goals in the context of resource sharing via an indistinguishability game. We also propose Bazaar, a protocol that allows participants to find suitable matches while satisfying our formal privacy goals.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**; **Privacy-preserving protocols**; *Security requirements.*

KEYWORDS

resource sharing, anonymous communication

ACM Reference Format:
Christoph Coijanovic, Daniel Schadt, Christiane Weis, and Thorsten Strufe. 2023. Bazaar: Anonymous Resource Sharing. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society (WPES '23), November 26, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3603216.3624957

1 INTRODUCTION

In most situations where a limited resource is shared by multiple parties, the resource is unevenly distributed. Consider manufacturing: Each company has a fixed capacity, defined by, e.g., the number of workers or assembly lines. At the same time, the demand for the company's goods fluctuates, depending on, for example, the economic situation or the life cycle of its product. As a result, some companies may have less to produce than they can handle, while others may have more. If companies could share their infrastructure, overall efficiency would improve, resulting in less environmental impact and lower prices for consumers. Note that this sharing of manufacturing capacity can be observed in the real world, such as between Toyota and Fuji Heavy Industries (FHI), where the latter manufactured cars for the former.[1] Other examples include distributed power grids (excess electricity stored in batteries can be sold to consumers) and logistics (several companies can share space on a container ship).

However, participating in a marketplace to find other parties with whom to share resources carries significant privacy risks: If a party can be linked to its current resource usage, others can use this information to its detriment. For example, energy consumption can be used to learn sensitive private information about users, such as what TV program they are watching [14]. Even if parties cannot be directly linked to their usage of the resource, the distribution of usage values can still reveal sensitive information. On the one hand, values from the distribution can be linked back to participants with background knowledge: If Alice has the largest photovoltaic system in her neighborhood, she is likely to have the most power to spare and can be easily identified. On the other hand, even without background knowledge, the distribution of usage values can reveal sensitive information: If a manufacturing company learns that *some* competitor is doing much worse than the others, it might lower prices to try to drive them out of business.

In this paper, we approach this challenging problem in two steps. First, we formalize privacy in resource sharing using an indistinguishability game. We propose two notions of privacy, one that formalizes that participants should not be linkable to their resource usage, and one that formalizes that the distribution of resource usage values should be hidden. The formal goals allow us to then propose Bazaar, a resource sharing protocol that provably achieves these goals.

For optimal matching, all participants provide their resource usage information to a central third party. Since we assume that the third party is honest-but-curious, we need to disassociate participants from their information and obfuscate the information itself. To do this, participants generate *tokens*, which together represent how much of the resource they have to spare or need. The tokens are sent to the third party through an anonymous channel, which ensures that the third party cannot tell how many tokens it receives from whom. The third party matches tokens indicating a deficit with tokens indicating a surplus and publishes the result. Finally, token holders are given the ability to anonymously contact the

[1] https://global.toyota/en/newsroom/corporate/25607796.html – Accessed September 26, 2023

WPES '23, November 26, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0235-8/23/11…$15.00
https://doi.org/10.1145/3603216.3624957

party with whom they have been matched via a temporary shared address.

In summary, this paper formalizes privacy in resource sharing and proposes Bazaar, which allows resource sharing with central matching without revealing the link between participants and their resource usage, nor the distribution of resource usage values.

2 RELATED WORK

Resource sharing can be expressed as a *linear programming* problem, where an objective function (total usage) is to be maximized while respecting a set of constraints (the surplus/deficit of the parties involved). There is a line of work [13, 6] that aims to solve these linear programming problems in a privacy-preserving manner. However, these protocols are based on *outsourcing*, where a client has access to the entire problem instance but does not have the computational resources to solve it [10]. In our setting, there is no client with a global view of all constraints, which makes these solutions inapplicable.

Matching. Another line of research [5, 8] aims to make *matching* algorithms private. The goal of these algorithms is to match each participant with another eligible participant. This 'one to one' matching however makes these protocol not optimal for our usecase. We want to be able to match a party to several others who, in sum, can satisfy the first party's needs.

Ride-Sharing. Privacy preserving ride-sharing [7, 12, 1] is a specific but related problem to our work. There, drivers and passengers are matched to form optimal routes. However, these protocols focus on hiding the exact location of passengers from the system, which may be applicable to some use cases of our system (e.g., logistics), but not others.

3 MODEL & GOALS

3.1 Setting and Adversary

We assume a set of agents $Ag = \{a_0, \dots, a_\ell\}$ acting on behalf of an entity. All entities are interested in a common resource. For brevity, we define a *resource usage function* that maps agents to 'their' usage of the resource:

$$RU(a : Ag) \mapsto \mathbb{Z}$$

If agent a's resource usage is negative ($RU(a) < 0$), we say that a has a *deficit* of the resource (i.e., wants more than it has). If the resource usage of agent a is positive ($RU(a) > 0$), we say that a has a *surplus* of the resource (i.e., has more than it uses). Surplus and deficit agents are matched by an honest but curious third party P.

Applied to our introductory example of Toyota and FHI, the setting works as follows: Toyota and FHI are both entities that each send an agent to find a match on their behalf. The common resource in this case is car manufacturing capacity. Agents from other automakers can participate at the same time. Since Toyota cars are produced in an FHI plant, Toyota's agent reports a deficit, while FHI's agent reports a surplus. The actual resource use could be expressed as the number of cars produced per day.

3.2 Privacy Goals

To allow for a rigorous privacy analysis, we need to formalize Bazaar's privacy goals. As mentioned in the introduction, we informally aim to achieve the following two goals:

(1) *Agent-Usage Unlinkability* ($AU\bar{L}$). Given the set of participating agents and the distribution of usage values, it should not be possible to link agents to their resource usage.
(2) *Usage Unobservability.* ($U\bar{O}$). Given the view of the third party P, it should not be possible to determine the real distribution of resource usage values.

We formalize both goals using a common indistinguishability game, similar to formalization approaches for message confidentiality [2] and anonymous communication [9].

Privacy Game. The game is played between a challenger C and an adversary $\mathcal{A}$. The adversary constructs a challenge $Ch = (S_0, S_1)$ consisting of two scenarios. Each scenario contains tuples $(a, RU(a), d)$ that define the challenge agent a, their usage $RU(a)$, and the data $d \in \{0,1\}^*$ to be exchanged with their match(es) (e.g., out-of-band contact information). The challenger randomly selects a scenario and internally executes the matching protocol (Bazaar in our case) on the contained input data. The adversary receives any protocol output and has to decide which of its scenarios was chosen by the challenger. Specifically, the game proceeds as follows:

(1) C draws a challenge coin $b \in \{0,1\}$ uniformly at random.
(2) $\mathcal{A}$ submits a challenge $Ch = (S_0, S_1)$.
(3) C executes the protocol with input S_b and sends any protocol output $\mathcal{A}$ would be able to observe in a 'real' run to $\mathcal{A}$.
(4) $\mathcal{A}$ processes the received information and returns its guess $b' \in \{0,1\}$.

Steps 1 and 2 can be repeated multiple times to allow $\mathcal{A}$ to adapt its behavior based on the observed output. $\mathcal{A}$ wins the game if $b = b'$.

Agent-Usage Unlinkability ($AU\bar{L}$). If a protocol reveals *any* information that depends on the input, $\mathcal{A}$ can use it to identify the chosen scenario and win the game above. If we only want to test whether the protocol reveals the link between agents and their usage, we must ensure that the adversary can only submit challenges where all other information is identical in both scenarios (and therefore cannot be used to distinguish).

To ensure that the adversary obeys this restriction, we extend step 2 of the game as follows:

2. $\mathcal{A}$ submits a challenge $Ch = (S_0, S_1)$. C checks if
   (a) $|S_0| = |S_1|$
   (b) Let $(a, RU(a), d)$ be the ith tuple in S_0, then $(a', RU(a), d)$ must be the ith tuple in S_1 for all $i \in \{0, \dots, |S_0| - 1\}$
   (c) Require

$$\forall (a, RU(a), d_a), (b, RU(b), d_b) \in S_b : a \neq b$$
$$\forall (a, RU(a), d_a) \in S_0 : (a, RU(a)', d'_a) \in S_1$$

   If any of theses checks fail, C rejects the challenge and instructs $\mathcal{A}$ to submit a new one.

The protocol achieves Agent-Usage Unlinkability, if there exists no probabilistic poly-time adversary who can win this restricted game with a non-negligible advantage over random guessing.

Usage Unobservability ($U\bar{O}$). Similarly, for Usage Unobservability, the challenger must ensure that the scenarios differ only in their distribution of resource usage values. Thus, the following conditions must hold in step 2:

(a) $|S_0| = |S_1|$
(b) Let $(a, RU(a), d)$ be the ith tuple in S_0, then $(a, RU(a)', d)$ must be the ith tuple in S_1 for all $i \in \{0, \dots, |S_0| - 1\}$
(c) Require $\forall (a, RU(a), d_a), (b, RU(b), d_b) \in S_b : a \neq b$
(d) Require

$$\sum\left\{RU(a)\,\middle|\,\begin{matrix}(a, RU(a), d) \in S_0\\ RU(a) > 0\end{matrix}\right\} = \sum\left\{RU(a)\,\middle|\,\begin{matrix}(a, RU(a), d) \in S_1\\ RU(a) > 0\end{matrix}\right\}$$
$$\sum\left\{RU(a)\,\middle|\,\begin{matrix}(a, RU(a), d) \in S_0\\ RU(a) < 0\end{matrix}\right\} = \sum\left\{RU(a)\,\middle|\,\begin{matrix}(a, RU(a), d) \in S_1\\ RU(a) < 0\end{matrix}\right\}$$

Without restriction (d), the adversary could trivially distinguish scenarios based on the ration of surplus to deficit.

4 DESIGN

Consider the following naive resource sharing protocol: All agents send their resource usage value along with an identifier (e.g., an email address) directly to the third party $\mathcal{P}$, which runs the matching algorithm and publishes the results. Agents can look up the identifiers of their new partners and contact them. We observe three main challenges in making the naive protocol anonymous:

(1) Agents need to send their values to $\mathcal{P}$ such that $\mathcal{P}$ does not learn which value is from whom.
(2) Values have to be obscured such that $\mathcal{P}$ does not learn the distribution of real values.
(3) Identifiers have to be anonymous such that $\mathcal{P}$ cannot determine who it has matched to whom while still allowing agents to contact each other.

Bazaar solves these challenges as follows:

Unlinking Agents and Values. Bazaar relies on an anonymous communication channel Ch_{anon} to unlink agents from the information they send to $\mathcal{P}$. Formally, we need Ch_{anon} to achieve Kuhn et al.'s notion of Sender Unobservability ($S\bar{O}$) against a corrupted receiver [9]. Informally, achieving $S\bar{O}$ means that no information about senders is revealed (to corrupted receivers in our case). Note that we also need Ch_{anon} to have an *anonymous reply* feature, where the receiver can send a message back to the sender without learning their identity. Ch_{anon} is used for all communication between agents and $\mathcal{P}$, and can be implemented against a corrupted recipient by Tor [3] or Loopix [11]. Note that while Tor achieves this protection if the adversary is only able to corrupt receivers, it does not achieve it against other adversaries such as global passive observers.

Obscuring Values. Instead of sending their usage values directly to $\mathcal{P}$, agents generate *tokens*. Each token is of either 'surplus' or 'deficit' type and corresponds to a *unit* of the resource (e.g., shipping containers, Wh of energy). The sum of the tokens an agent sends to $\mathcal{P}$ corresponds to its usage value. The privacy of Ch_{anon} ensures that $\mathcal{P}$ cannot determine how many tokens an agent has sent.

Anonymous Identifiers and Agent Contact. To allow anonymous contact between agents after a match, agents attach a random ephemeral public key to each token. The output of $\mathcal{P}$ is a list of pairs of these public keys, indicating which token has been matched to which. Matched agents derive a common identifier from their public keys. To exchange messages, both agents send a tuple of this identifier and their message via Ch_{anon} to $\mathcal{P}$. $\mathcal{P}$ matches the identifier and uses the reply function of Ch_{anon} to send each agent the other agent's message.

Figure 1 visualizes the resulting protocol.

5 PROTOCOL

We will now describe our Bazaar protocol in more detail. Each Bazaar run consists of three phases: *Token Accumulation*, *Token Matching*, and *Agent Coordination*.

Token Accumulation. The goal of the token accumulation phase is for all agents to submit their usage information to $\mathcal{P}$. Let agent a have usage $RU(a)$. After the start of a new run is announced by $\mathcal{P}$, a generates exactly $RU(a)$ tokens. For each token t, a does the following:[2]

$$(pk_t, sk_t) \leftarrow \textsc{KeyGen}(1^\lambda)$$
$$\texttt{type}_t \leftarrow \begin{cases} + & \text{if } RU(a) > 0 \\ - & \text{else} \end{cases}$$
$$\texttt{sig}_t \leftarrow \textsc{Sign}(sk_t, pk_t \parallel \texttt{type})$$
$$t \leftarrow (pk_t, \texttt{type}_t, \texttt{sig}_t)$$

Agents may need to generate many tokens, each with a new key pair. To reduce the computational overhead of key generation, it can be optimized based on the encryption scheme used. For example, if ElGamal encryption [4] is used, agents only need to generate a single cyclic group G of order q with generator g. Then, for each token, the agent chooses a uniformly random integer $x \in \{1, \dots, q-1\}$ as the secret key and computes the public key (G, g, q, g^x) with a single exponentiation.

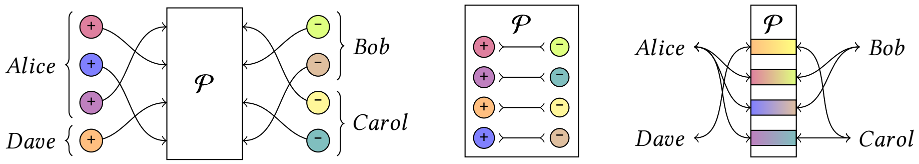
After the tokens are generated, a sends each token *individually* via Ch_{anon} to $\mathcal{P}$. $\mathcal{P}$ collects all incoming tokens, verifies the signatures they contain (using the token's public key), and stores them for later use. Recall that we consider a local adversary at $\mathcal{P}$. To protect against a global adversary, dummy tokens would need to be introduced to fix the number of tokens each agent sends.

Token Matching. The goal of the token matching phase is for $\mathcal{P}$ to find a partner for each token and publish the results. First, $\mathcal{P}$ divides the tokens into two lists, each sorted lexicographically by its tokens' public keys: $S = \{t_{s,0}, t_{s,1}, \dots\}$ contains all tokens of type 'surplus' (+) and $D = \{t_{d,0}, t_{d,1}, \dots\}$ contains all tokens of type 'deficit' (-). Then $\mathcal{P}$ *zips* the lists S and D into a new list $R := \{(t_{s,0}, t_{d,0}), (t_{s,1}, t_{d,1}), \dots\}$. R is sent to all agents.

Agent Coordination. In the agent coordination phase the two agents belonging to any matched token pair will exchange a message with each other. If multiple messages have to be exchanged, this phase can be repeated. Assume agent a with token $t_a = (pk_{t_a}, +, \texttt{sig}_{t_a})$ and agent b with token $t_b = (pk_{t_b}, -, \texttt{sig}_{t_b})$ have been matched. To send message m to b, a proceeds as follows:

(1) Verify that $\texttt{sig}_{t_b}$ is valid. If not, abort.
(2) $c_a \leftarrow \textsc{Enc}(pk_{t_b}, m)$
(3) $\texttt{addr} \leftarrow H(pk_{t_a} \parallel pk_{t_b})$ where $H(\cdot)$ is a hash function.

[2] We use "$a \parallel b$" to denote the concatenation of a and b



1. Token Accumulation 2. Token Matching 3. Agent Coordination

Figure 1: Protocol phases of Bazaar. Arrows indicate communication over an anonymous channel. In the first phase, agents send tokens representing their resource usage to the third party $\mathcal{P}$. In the second phase, $\mathcal{P}$ matches tokens indicating a surplus with those indicating a deficit. In the third phase, agents whose tokens have been matched exchange messages with each other.

(4) $\texttt{sig}_a \leftarrow \textsc{Sign}(sk_a, \texttt{addr} \parallel c_a)$
(5) Send packet $p_a := (\texttt{addr}, c_a, \texttt{sig}_a)$ through Ch_{anon} to $\mathcal{P}$

After collecting all incoming packets, $\mathcal{P}$ iterates through them. For every found pair $p = (\texttt{addr}, c, \texttt{sig}), p' = (\texttt{addr}, c', \texttt{sig}')$ with matching addresses, it uses Ch_{anon}'s anonymous reply feature to send $(c', \texttt{sig}')$ to the sender of p and $(c, \texttt{sig})$ to the sender of p'. Upon receiving a reply, the agents verify that the signature is valid using the matching agent's public key, and use their secret keys to decrypt the ciphertext revealing the matched agent's message.

6 SECURITY ANALYSIS

In this section, we argue that Bazaar does indeed achieve the privacy goals defined in Section 3.2. Recall that we consider an honest but curious $\mathcal{P}$ as the adversary.

Theorem 1. *Bazaar achieves $AU\bar{L}$.*

Proof. (*Sketch*) To break $AU\bar{L}$, $\mathcal{A}$ must be able to distinguish between two scenarios that differ only in which agent has which usage. Assume the existence of such a $\mathcal{A}$. $\mathcal{A}$ can be used by an adversary $\mathcal{B}$ to break the assumed $S\bar{O}$ protection of Ch_{anon}. For a description of $S\bar{O}$ refer to Appendix A. $\mathcal{B}$ must provide $\mathcal{A}$ with the observations that $\mathcal{P}$ can make in both the token accumulation and agent coordination phases.

Let $(a, RU(a), d)$ be the ith tuple in $\mathcal{A}$'s scenario 0. If $\mathcal{A}$'s challenge is *not* valid for $AU\bar{L}$, $\mathcal{B}$ rejects it. Otherwise, the ith tuple of $\mathcal{A}$'s scenario 1 is by definition $a', RU(a), d$.

$\mathcal{B}$ first constructs a $S\bar{O}$ challenge corresponding to the token accumulation phase. From the ith tuples of $\mathcal{A}$'s challenge, $\mathcal{B}$ derives $RU(a)$ communications for each scenario as follows:

Scenario 0	**Scenario 1**
$(a, t_1, \mathcal{P})$	$(a', t_1, \mathcal{P})$
...	...
$(a, t_{RU(a)}, \mathcal{P})$	$(a', t_{RU(a)}, \mathcal{P})$

The messages $t_1, \ldots, t_{RU(a)}$ are constructed by $\mathcal{B}$ as a real agent in Bazaar would construct a token: For each t_j, a new key pair is chosen, a type is determined based on the sign of $RU(a)$, and a signature is generated over the public key and type using the secret key. The communications derived from all tuples of $\mathcal{A}$ are concatenated and submitted to C as a $S\bar{O}$ challenge. C selects a scenario at random, executes the contained communication, and sends to $\mathcal{B}$ the observations that a corrupted receiver $\mathcal{P}$ can make. $\mathcal{B}$ forwards these observations to $\mathcal{A}$.

To provide the observations for the agent coordination phase, $\mathcal{B}$ derives another $S\bar{O}$ challenge from the challenge of $\mathcal{A}$. The second challenge is derived analogously to the first, but instead of tokens, d is used as the message for communication.

$\mathcal{A}$'s observations match those it could make by interacting with the real protocol, since 1) $\mathcal{B}$ generates the messages as the agents would, and 2) in the real protocol, $\mathcal{P}$ also receives all communications through Ch_{anon}, which is simulated by C. If $\mathcal{A}$ can distinguish based on $\mathcal{B}$'s output, $\mathcal{B}$ can in turn determine which scenario was chosen by the $S\bar{O}$ challenger. □

Theorem 2. *Bazaar achieves $U\bar{O}$.*

Proof. (*Sketch*) We show that any information $\mathcal{P}$ can observe is independent of the distribution of usage values. Due to the restriction of $U\bar{O}$ that the sums of positive and negative usage values must be identical in both scenarios, $\mathcal{P}$ receives the same number of surplus and deficit tokens in both scenarios. As described in Section 5, the tokens are generated independently. What differs based on the distribution of values is who sends how many tokens. However, this information is hidden by the assumed $S\bar{O}$ of Ch_{anon}. By the same logic, it can be argued that the observations in the agent coordination phase are independent of the distribution of usage values. □

7 CONCLUSION

In this paper, we have introduced Bazaar, which allows resource sharing with matching by an honest-but-curious third party with strong privacy. We have formalized the privacy goals of Agent-Usage Unlinkability and Resource Usage Unobservability based on a common indistinguishability game and have shown that Bazaar achieves both. Our formal privacy goals not only enable the analysis of Bazaar, but can also be used by future protocols to allow fair comparisons of the privacy protection of different approaches. Finally, future work includes the consideration of stronger adversaries, alternative ways to obfuscate usage values, multi-resource scenarios, and an empirical evaluation of the proposed protocol.

Acknowledgements. This work has been funded by the Helmholtz Association through the KASTEL Security Research Labs (HGF Topic 46.23), and by funding of the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden.

REFERENCES

[1] U. Aïvodji et al. 2018. Sride: a privacy-preserving ridesharing system. *ACM WISEC*.

[2] Mihir Bellare et al. 1998. Relations among notions of security for public-key encryption schemes. In *IACR Cryptology ePrint Archive*.

[3] Roger Dingledine et al. 2004. Tor: the second-generation onion router. In *USENIX Security*.

[4] Taher Elgamal. 1984. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Annual International Cryptology Conference*.

[5] Philippe Golle. 2006. A private stable matching algorithm. In *Financial Cryptography*.

[6] Yuan Hong and Jaideep Vaidya. 2014. An inference–proof approach to privacy-preserving horizontally partitioned linear programs. *Optimization Letters*.

[7] Junxin Huang, Yuchuan Luo, Ming Xu, Bowen Hu, and Jian Long. 2022. Pshare: privacy-preserving ride-sharing system with minimum-detouring route. *Applied Sciences*.

[8] Andreas Klinger and Ulrike Meyer. 2023. Privacy-preserving fully online matching with deadlines. *ACM CODASPY*.

[9] Christiane Kuhn et al. 2018. On privacy notions in anonymous communication. *PoPETS*.

[10] Peeter Laud and Alisa Pankova. 2013. On the (im)possibility of privately outsourcing linear programming. *ACM CCSW*.

[11] Ania M. Piotrowska et al. 2017. The loopix anonymity system. *USENIX Security*.

[12] Sara Ramezanian et al. 2022. Lightweight privacy-preserving ride-sharing protocols for autonomous cars. *ACM CSCS*.

[13] Jaideep Vaidya. 2009. Privacy-preserving linear programming. In *ACM SIGAPP*.

[14] Liehuang Zhu et al. 2017. Privacy protection using a rechargeable battery for energy consumption in smart grids. *IEEE Network*.

A KUHN ET AL.'S SENDER UNOBSERVABILITY

Kuhn et al. define a set of privacy notions for anonymous communication [9]. One of these notions is *Sender Unobservability*, abbreviated as $S\bar{O}$, which intuitively expresses that the communication protocol may disclose information about active receivers and messages, but has to hide all information about senders.

Sender Unobservability is formalized via an indistinguishability game played between a challenger C and an adversary $\mathcal{A}$. Each *communication* is expressed as a tuple (s, r, m, aux), where s is the sender, r the receiver, m the message, and aux optional auxiliary information such as session identifiers. The lack of a communication is denoted with the symbol '$\diamond$'. Communications are grouped into *batches*, which in turn are grouped into *scenarios*.

The $S\bar{O}$ game proceeds as follows:

(1) C draws a coin $b \in \{0, 1\}$ uniformly at random.

(2) $\mathcal{A}$ submits a challenge $Ch = (S_0, S_1)$ consisting of two scenarios.

(3) Let $\underline{r}_{b,i}$ be the ith batch of scenario S_b containing communications $r_{b,i,j} \in \{(s_{b,i,j}, r_{b,i,j}, m_{b,i,j}, aux_{b,i,j}), \diamond\}$. C checks if for all $b \in \{0, 1\}$, $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, l\}$:

$$\begin{aligned} r_{1,i,j} &= (s_{1,i,j}, r_{0,i,j}, m_{0,i,j}, aux_{0,i,j}) \\ \diamond &\notin \underline{r}_{b,i} \end{aligned}$$

This check ensures that communications between scenarios only differ in their senders and that there are no 'empty' communications in either scenario.

(4) C simulates the protocol to be tested Π with the communications from S_b.

(5) Any protocol output $\Pi(S_b)$ during this simulation is forwarded to $\mathcal{A}$.

(6) $\mathcal{A}$ processes the received information and returns its guess $b' \in \{0, 1\}$ for b.

$\mathcal{A}$ wins if $b' = b$. Π achieves $S\bar{O}$ if there exists no probabilistic polynomial time adversary who can win the game defined above with a non-negligible advantage over random guessing.

Received 07/24/2023; accepted 09/13/2023