Secure Best Arm Identification in Multi-armed Bandits

Radu Ciucanu[1](✉), Pascal Lafourcade[2], Marius Lombard-Platet[3,4], and Marta Soare[1]

[1] INSA Centre Val de Loire, Univ. Orléans, LIFO EA 4022, Orléans, France
radu.ciucanu@insa-cvl.fr, marta.soare@univ-orleans.fr
[2] Université Clermont Auvergne, LIMOS CNRS UMR 6158, Aubière, France
pascal.lafourcade@uca.fr
[3] Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, Paris, France
marius.lombard-platet@ens.fr
[4] Be-Studys, Geneva, Switzerland

Abstract. The stochastic multi-armed bandit is a classical decision making model, where an agent repeatedly chooses an action (pull a bandit arm) and the environment responds with a stochastic outcome (reward) coming from an unknown distribution associated with the chosen action. A popular objective for the agent is that of identifying the arm with the maximum expected reward, also known as the *best-arm identification* problem. We address the inherent privacy concerns that occur in a best-arm identification problem when outsourcing the data and computations to a *honest-but-curious* cloud.

Our main contribution is a distributed protocol that computes the best arm while guaranteeing that (i) no cloud node can learn at the same time information about the rewards and about the arms ranking, and (ii) by analyzing the messages communicated between the different cloud nodes, no information can be learned about the rewards or about the ranking. In other words, the two properties ensure that the protocol has no security single point of failure. We rely on the partially homomorphic property of the well-known Paillier's cryptosystem as a building block in our protocol. We prove the correctness of our protocol and we present proof-of-concept experiments suggesting its practical feasibility.

Keywords: Multi-armed bandits · Best arm identification · Privacy · Distributed computation · Paillier cryptosystem

1 Introduction

In a stochastic multi-armed bandit model, a learning agent sequentially needs to decide which *arm* (option/action) to pull from K arms with unknown associated

This project is partially funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 826404.

S.-H. Heng and J. Lopez (Eds.): ISPEC 2019, LNCS 11879, pp. 152–171, 2019.
https://doi.org/10.1007/978-3-030-34339-2_9

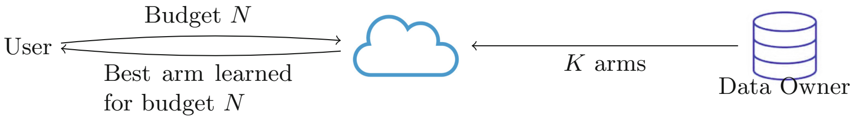


Fig. 1. System architecture.

values available in the learning environment. After each pull, the environment responds with a feedback, in the form of a stochastic *reward* from an unknown distribution associated with the arm chosen by the agent. This is a dynamic research topic with a wide range of applications, including clinical trials for deciding on the best treatment to give to a patient [17], on-line advertisements and recommender systems [12], or game playing [4,11,14].

In this paper, we focus on a popular objective in multi-armed bandits, that of *best arm identification*: given a set of K arms and a limited budget of N pulls, the goal of the agent is to design a budget-allocation strategy that maximizes the probability of identifying the arm with the maximum expected reward. This problem has been extensively studied in the machine learning community [1,3,7,8,10,16], but to the best of our knowledge, there is no previous work that considers this problem from a privacy-preserving viewpoint. Next, we illustrate the problem via a motivating example.

Use Case Example. A classical real-world application of the *best-arm identification* problem is as follows. Before launching a new product on the market, companies can create several versions of the product that are put into a testing phase. By *product*, we refer here to any type of object/service that might be offered by a company and that may contain (or be obtained as a result of analyzing) private data. Each version of the product has distinguishing characteristics and the company surveys potential customers about the version they prefer. The company's objective is that once the testing phase is over, it can put on the market the version that is likely to yield the best sales. The goal of the best-arm identification problem is to define algorithms that maximize the probability of identifying the best arm (here, the best version among the K alternative versions), given a limited budget of N observations (here, customer surveys). Therefore, best-arm identification algorithms are a good fit for the product testing phase.

Now, imagine the scenario where a company collected over the years a large quantity of customer surveys that it no longer needs for its purposes. This data may actually be useful for other smaller companies that cannot afford doing their own customer surveys, but nonetheless want to simulate the test of different versions of their product. This brings us to the system architecture depicted in Fig. 1. The *data owner* is the company that owns a large quantity of customer surveys that it wants to monetize. The *user* is the small company that wants to simulate the testing of different versions of its product, without conducting its own customer survey. It may actually be cheaper to pay a limited budget to reuse pieces of existing data, rather than doing a new survey with real customers.

The interaction between the data owner and the user is done using some *public cloud*, where initially the data owner outsources its data, then the users interact directly with the cloud. More precisely, each user allocates some budget to the cloud and reuses the available surveys for deciding which version of its own product should be put on the market. The budget would refer here to the number of survey answers the user wants the cloud to use before outputting the best option. As a simplified example, assume that the available data consists of user preferences about the characteristics of security devices they would buy for protecting their homes. There are 1M surveys available and consulting a survey costs 0.1$. If a small company wants to know which type of device people from their market are more likely to buy, the precision of the answer it receives from the cloud depends on the paid budget. If it pays 100$, it is more likely to get a clearer image about the type of device that is the most likely to be purchased, than if it pays 5$. But in both cases the obtained information is not 100% sure because only a sample of the available data is consulted.

The aforementioned use case can be easily reformulated to other real-world scenarios, such as health or medical data, cosmetics (e.g., trials for finding the best anti-wrinkle cream), data concerning political preferences, education and employment records, to name a few.

As already mentioned, we consider a scenario where the multi-armed bandits (i.e., the *data*) as well as the best arm identification algorithm (i.e., the *computation*) are outsourced to some public cloud. We assume that the cloud is *honest-but-curious*: it executes tasks dutifully, and try to gain information on the ranking of the arms and their associated values from the data they receive. We address the privacy concerns that occur when outsourcing the data and computations.

Indeed, the externalized data can be communicated over an untrustworthy network and processed on some untrustworthy machines, where malicious public cloud users may learn private data that belongs only to the data owner. This is why we require the data owner to encrypt all information about the arms before outsourcing the data to the public cloud.

Moreover, each cloud user observes a result of the best arm identification algorithm that is proportional to the budget that the user pays. It should be impossible for a malicious cloud user to compose observations of several runs of the best arm identification algorithm in order to learn the best arm with a higher confidence, and then sell this information to some other user.

Summary of Contributions and Paper Organization. In Sect. 2, we give background information on the problem of best-arm identification in multi-armed bandits.

In Sect. 3, we first present the considered security model, then the needed security tools, and finally the distributed security protocol, that is the main contribution of the paper. We rely on the partial homomorphic property of Paillier's cryptosystem [15] as a building block in our protocol. The difficulty of our setting comes from the fact that we need additions, multiplications, and comparisons to solve the best arm identification problem, whereas a partially homomorphic cryptosystem such as Paillier's provides only homomorphic additions. Therefore, in our protocol we distribute the computation among several participants and insure that each of them can only learn the specific information needed for performing their task, and no any other information. Thus, we show that our distributed protocol has no single point of failure, in the *honest-but-curious* cloud model. The overhead due to the security primitives of our protocol depends only on the number of arms K and not on the budget N. This is a desirable property because in practice the budget N (i.e., the number of arm pulls that we are allowed to do) is often much larger than the number of arms K among which we can choose.

We prove the security of our protocol in Appendix A, and we present proof-of-concept experiments suggesting its feasibility in Sect. 4. We discuss related work in Sect. 5, and conclusions and future work in Sect. 6.

2 Primer on Multi-armed Bandits

The problem of *best arm identification in multi-armed bandits* [1] has been initially formulated in the domain of real numbers. We slightly revisit the initial formulation of the problem in order to manipulate integers. The reason behind this adaptation is that later on in our protocol, we rely on public key cryptography tools to add security guarantees to a state-of-the-art best arm identification algorithm.

Input. The input is twofold:

- Number of arms K. Each arm $i \in \{1, \ldots, K\}$ is associated to a reward value $x(i)$ and a reward function r that returns a random integer in an interval $[x(i) - \epsilon, x(i) + \epsilon]$ according to a uniform probability distribution. Whereas each arm i is associated to its specific value $x(i)$, the value of ϵ is common to all arms. The intervals associated to different arms may be overlapping, which makes the setting non-trivial. The best arm i^* is $\arg\max_{i \in \{1,\ldots,K\}} x(i)$.
- Budget N that means how many arm pulls (and implicit reward observations) the user is allowed to do.

Note that for designing a budget-allocation strategy between the arms, only the number of arms K and the budget N are known. There is no initial information about the reward associated to each arm.

Output. The estimated best arm $\widehat{i^*}_\alpha$ that can be learned after making N arm pulls (and subsequent reward observations) according to some allocation strategy α, which defines how the budget is divided between the K arms. The challenge is to design a budget-allocation strategy α that makes the best possible use of the budget. In other words, when selecting the arms to be pulled according to α, the observed rewards allow to acquire as much useful information as possible for identifying i^*.

Performance Measure. We call *simple regret* R_N the difference between the value of the (true) best arm i^* and the arm $\widehat{i^*}_\alpha$ estimated as being the best arm by an allocation strategy α after N arm pulls. Thus, we compare the gap between the value of the identification made by strategy α and that of an *oracle* strategy that knows the values of the arms beforehand. Formally, the performance of strategy α after using a budget N is $R_N(\alpha) = x(i^*) - x(\widehat{i^*_\alpha})$.

Example. We have 3 arms with associated reward values in intervals [3, 23], [25, 45], and [40, 60]. This means that $x(1) = 13, x(2) = 35, x(3) = 50$, and $\epsilon = 10$. Assuming a budget of 3, the user may choose to spend one pull for each arm and observe rewards of (for instance) 23, 44, and 41, respectively. Hence, the user could wrongly think that arm 2 is the best, thus getting a regret of $50 - 35 = 15$.

Obviously, increasing the budget would increase the number of pulls that can be done, hence it would increase the chances of correctly identifying the best arm. This can be easily done in the presence of an infinite budget, but the challenge is to identify the best arm using as few pulls as possible, or in other words, to maximize the probability of correctly identifying the best arm while having a limited budget.

Successive Rejects (`SR`) [1]. The algorithm takes as input the number of arms K and the budget N. Initially, all K arms are candidates. `SR` divides the budget in $K-1$ phases. At the end of each phase, a decision is made. The phases' lengths are fixed such that the available budget is not exceeded and the probability of wrongly identifying the best arm is minimized.

More precisely, at each phase $j \in \{1, \ldots, K-1\}$, each still candidate arm in A_j is pulled n_j times according to the fixed allocation (cf. Algorithm 1). At the end of each phase, the algorithm rejects the arm with the lowest sum of observed rewards, that is the arm estimated to be the worst. If there is a tie, `SR` randomly selects the arm to reject among the worst arms. Then, at the next phase, the remaining arms are again uniformly pulled according to the fixed allocation. Thus, the worst arm is pulled n_1 times, the second worst is pulled $n_2 + n_1$ times, and so on, with the best and the second-best arm being pulled $n_{K-1} + \ldots + n_1$ times. The estimated best arm is the unique arm remaining after phase $K-1$.

We consider the sums of observed rewards per arm when deciding which arm to reject instead of empirical means as in the original version [1] as a simplification. Indeed, each candidate arm is pulled the same number of times in each phase, hence the ranking of the arms is identical regardless of whether we look at sums or means.

Example. Let a multi-armed bandit with 4 arms and $x(1) > x(2) > x(3) > x(4)$, with budget $N = 500$ pulls. We have $\overline{log}(4) = \frac{1}{2} + \sum_{i=2}^{4} \frac{1}{i} = \frac{19}{12}$ and:

Phase 1: each arm 1, 2, 3, 4 is pulled $n_1 = \left\lceil \frac{12}{19} \frac{500-4}{4+1-1} \right\rceil = 79$ times
Phase 2: each arm 1, 2, 3 is pulled $n_2 = \left\lceil \frac{12}{19} \frac{500-4}{4+1-2} \right\rceil - n_1 = 26$ times
Phase 3: each arm 1, 2 is pulled $n_3 = \left\lceil \frac{12}{19} \frac{500-4}{4+1-3} \right\rceil - (n_1 + n_2) = 52$ times.

In other words, arm 4 is pulled 79 times, arm 3 is pulled $79 + 26 = 105$ times, each arm 1, 2 is pulled $79 + 26 + 52 = 157$ times, totalling $79 + 105 + 2 \times 157 = 498$ pulls.

Algorithm 1. SR algorithm (adapted from [1])

```
$A_1 \leftarrow \{1, \ldots, K\}$                                   ▷ Initialization
for all $i \in A_1$ do
    $sum[i] \leftarrow 0$
$\overline{\log}(K) \leftarrow \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$
$n_0 \leftarrow 0$

for $j$ from 1 to $K - 1$ do                                       ▷ Successive rejects
    $n_j \leftarrow \left\lceil \frac{1}{\overline{\log}(K)} \frac{N-K}{K+1-j} \right\rceil - \sum_{l=0}^{j-1} n_l$
    for all $i \in A_j$ do
        loop $n_j$ times
            $r \leftarrow$ random integer from $[x(i) - \epsilon, x(i) + \epsilon]$
            $sum[i] \leftarrow sum[i] + r$
    $A_{j+1} \leftarrow A_i \backslash \arg\min_{i \in A_j} sum[i]$
return $A_K$
```

3 Secure Protocol

3.1 Security Model

We assume that the reward functions associated to the arms as well as the best arm identification algorithm are outsourced to some cloud. We assume that the cloud is honest-but-curious i.e., it executes tasks dutifully, but tries to extract as much information as possible from the data that it sees. The user indicates to the cloud her budget and receives the best arm that the cloud can compute using the user's budget. The user does not have to do any computation, except for eventually decrypting $\widehat{i^*}$ if she receives this information encrypted from the cloud. We expect the following security properties:

1. No cloud node can learn at the same time information about the rewards and about the ranking of the arms.
2. By analyzing the messages communicated between the different cloud nodes, no information can be learned about the rewards or about the ranking.

The two aforementioned properties essentially ensure that the desired protocol has no security single point of failure. In particular, the first property says that (i) there may be some cloud node that knows the ranking of the arms (hence also the best arm), but it is not allowed to know which rewards are associated to these arms, and (ii) there may also be some cloud node that knows some rewards, but it is not allowed to know which arms are associated to these rewards. If all cloud nodes collude, the cloud can learn the rewards associated to the arms[1]. We do not consider collusions in our model.

[1] In case of collusions, if several users spent successive budgets to learn the best arm among the same set of arms, the cloud could compose the observed rewards. Hence the cloud could compute the best arm using as budget the total budget of the users and leak this information to some malicious user.

3.2 Security Background

We use Pailler's public key encryption scheme [15]. We first recall the definition of public-key encryption. Pailler's encryption scheme is IND-CPA secure. We recall the definition of IND-CPA before presenting the scheme itself that has an additive homomorphic property that we use in our protocol.

Definition 1 (PKE). *Let η be a security parameter. A* public-key encryption *(PKE) scheme is defined by $(\mathcal{G}, \mathcal{E}, \mathcal{D})$:*

$\mathcal{G}(\eta)$: *returns a public/private key pair* $(\mathsf{pk}, \mathsf{sk})$.
$\mathcal{E}_{\mathsf{pk}}(m)$: *returns the ciphertext c.*
$\mathcal{D}_{\mathsf{sk}}(c)$: *returns the plaintext m.*

We also recall the notion of negligible function in order to define the IND-CPA security notion.

Definition 2. *A function $\gamma\colon \mathbb{N} \to \mathbb{N}$ is negligible in η, and is noted negl(η), if for every positive polynomial $p(\cdot)$ and sufficiently large η, $\gamma(\eta) < 1/p(\eta)$.*

Let $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a PKE scheme, $\mathcal{A}$ be a probabilistic polynomial-time adversary. For $b \in \{0, 1\}$, we define the IND-CPA-b experiment where $\mathcal{A}$ has access to the oracle $\mathcal{E}_{\mathsf{pk}}(LR_b(\cdot, \cdot))$ taking (m_0, m_1) as input and returns $\mathcal{E}_{\mathsf{pk}}(m_0)$ if $b = 0$, $\mathcal{E}_{\mathsf{pk}}(m_1)$ otherwise. $\mathcal{A}$ tries to guess the bit b chosen in the experiment. We define the advantage of $\mathcal{A}$ against the IND-CPA experiment by: $\mathsf{Adv}^{\text{ind-cpa}}_{\Pi,\mathcal{A}}(\eta) = \left|\Pr[1 \leftarrow \mathsf{Exp}^{\text{IND-CPA-1}}_{\Pi,\mathcal{A}}(\eta)] - \Pr[1 \leftarrow \mathsf{Exp}^{\text{IND-CPA-0}}_{\Pi,\mathcal{A}}(\eta)]\right|$. We said that Π is IND-CPA if this advantage is negligible for any probabilistic polynomial-time $\mathcal{A}$. Paillier's cryptosystem is an IND-CPA scheme. We give the key generation, the encryption and decryption algorithms.

Key Generation. We denote by $\mathbb{Z}_n$, the ring of integers modulo n and by $\mathbb{Z}_n^\times$ the set of invertible elements of $\mathbb{Z}_n$. The public key pk of Paillier's cryptosystem is (n, g), where $g \in \mathbb{Z}_{n^2}^\times$ and $n = pq$ is the product of two prime numbers such that $gcd(p, q) = 1$. The corresponding private key sk is (λ, μ), where λ is the least common multiple of $p - 1$ and $q - 1$ and $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$, where $L(x) = (x - 1)/n$.

Encryption Algorithm. Let m be a message such that $m \in \mathbb{Z}_n$. Let g be an element of $\mathbb{Z}_{n^2}^\times$ and r be a random element of $\mathbb{Z}_n^\times$. We denote by $\mathcal{E}_{\mathsf{pk}}(\cdot)$ the encryption function that produces the ciphertext c from a given plaintext m with the public key $\mathsf{pk} = (n, g)$ as follows: $c = g^m \cdot r^n \mod n^2$.

Decryption Algorithm. Let c be a ciphertext such that $c \in \mathbb{Z}_{n^2}^\times$. We denote by $\mathcal{D}_{\mathsf{sk}}(\cdot)$ the decryption function of c with the secret key $\mathsf{sk} = (\lambda, \mu)$ defined as follows: $m = L\left(c^\lambda \mod n^2\right) \times \mu \mod n$.

Paillier's cryptosystem is a partial homomorphic encryption scheme. Let m_1 and m_2 be two plaintexts in $\mathbb{Z}_n$. The product of the two associated ciphertexts

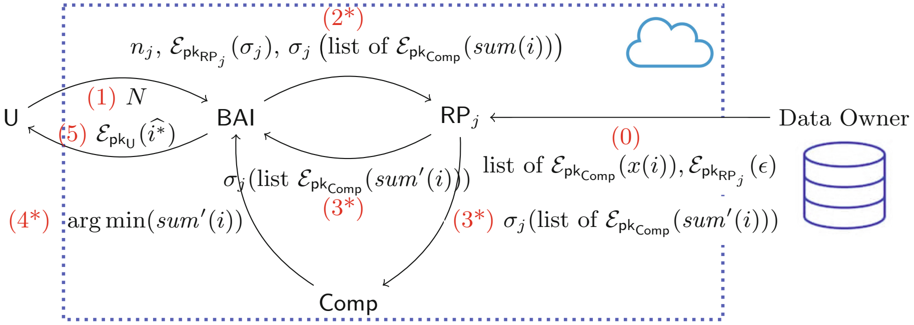

(2*)
n_j, $\mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j)$, σ_j (list of $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum(i))$)
(1) N
U
BAI
RP$_j$
Data Owner
(5) $\mathcal{E}_{\mathsf{pk}_{\mathsf{U}}}(\widehat{i^*})$
(0)
list of $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(x(i)), \mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\epsilon)$
σ_j(list $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum'(i))$)
(3*)
(4*) arg min$(sum'(i))$
(3*) σ_j(list of $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum'(i))$)
Comp

Fig. 2. Workflow of the secure algorithm. We use numbers to indicate the order of the steps. The steps annotated with * are repeated for each phase $j \in \{1, K-1\}$. For the communications BAI → RP$_j$, RP$_j$ → BAI, and RP$_j$ → Comp, the list concerns all the arms that are still candidates i.e., the set A_j.

with the public key $\mathsf{pk} = (n, g)$, denoted $c_1 = \mathcal{E}_{\mathsf{pk}}(m_1) = g^{m_1} \cdot r_1^n \mod n^2$ and $c_2 = \mathcal{E}_{\mathsf{pk}}(m_2) = g^{m_2} \cdot r_2^n \mod n^2$, is the encryption of the sum of m_1 and m_2, i.e., $\mathcal{E}_{\mathsf{pk}}(m_1) \cdot \mathcal{E}_{\mathsf{pk}}(m_2) = \mathcal{E}_{\mathsf{pk}}(m_1 + m_2 \mod n)$.

We also remark that: $\mathcal{E}_{\mathsf{pk}}(m_1) \cdot \mathcal{E}_{\mathsf{pk}}(m_2)^{-1} = \mathcal{E}_{\mathsf{pk}}(m_1 - m_2)$.

3.3 Secure Algorithm

We revisit the successive rejects (SR) algorithm in order to satisfy the properties outlined in Sect. 3.1. We consider K arms. We note $[\![n]\!]$ the set of the n first integers: $[\![n]\!] = \{1, \ldots, n\}$. Recall that SR has $K-1$ phases and at each phase j, it uses a budget of n_j to pull each of the still candidate arms. At the end of each phase, SR rejects the worst arm, based on all pulls observed since the beginning.

In the sequel, each time we refer to some $(\mathsf{pk}, \mathsf{sk})$, and associated encryption/decryption functions, we assume they are done using Paillier's cryptosystem [15]. In particular, we rely on the homomorphic addition property of Paillier's cryptosystem i.e., $\mathcal{E}_{\mathsf{pk}}(x + y) = \mathcal{E}_{\mathsf{pk}}(x) \cdot \mathcal{E}_{\mathsf{pk}}(y)$.

In our security protocol, we assume $K + 3$ participants:

- DO is the Data Owner, who is not in the cloud. DO sends the encrypted arm values $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(x_i)$ and $\mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\epsilon)$ for $i \in [\![K]\!]$ and $j \in [\![K-1]\!]$.
- U is the User, a participant that is not in the cloud. The user generates $(\mathsf{pk}_{\mathsf{U}}, \mathsf{sk}_{\mathsf{U}})$ and shares pk_{U} and the budget N with the cloud. The cloud nodes compute $\widehat{i^*}$ and at the end BAI sends $\mathcal{E}_{\mathsf{pk}_{\mathsf{U}}}(\widehat{i^*})$ to the user, who is able to decrypt it using her secret key sk_{U}.
- BAI (Best-Arm Identification) is the node responsible for executing the $K-1$ phases of the SR algorithm. BAI generates $K-1$ uniformly selected permutations σ_j of $[\![K+1-j]\!]$ (as there are $K+1-j$ candidate arms at round j). Each σ_j is shared with the node RP$_j$, but not with Comp. At each phase, BAI knows which arm is the worst and should be rejected, and after the last phase

Table 1. What each cloud node knows and does not know.

Node	BAI	Comp	RP$_j$
Does know	• ranking of arms (including best arm)	• sums of rewards	• arms still candidate at phase j • arms already rejected before phase j • sums of rewards added at phase j
Does not know	• sums of rewards of any arm (Theorem 1)	• mapping between sums of rewards and the arms that produced them (Theorem 3) • ranking of arms (including best arm) (Theorem 2)	• ranking of arms (including best arm) (Theorems 4 and 5) • sums of rewards from phases $1, \dots, j-1, j+1, \dots, K-1$ (Theorem 6)

it knows which arm is the best. However, BAI does not know which rewards are associated to the arms because the rewards are encrypted with $\mathsf{pk}_{\mathsf{Comp}}$.

– Comp is the node responsible of choosing the worst one among the sums of rewards associated to the candidate arms. Comp generates $(\mathsf{pk}_{\mathsf{Comp}}, \mathsf{sk}_{\mathsf{Comp}})$ and shares $\mathsf{pk}_{\mathsf{Comp}}$ with all other cloud nodes and DO.
– $\mathsf{RP}_1, \dots, \mathsf{RP}_{K-1}$ are $K-1$ nodes, each of them knowing the value ϵ that is needed to generate a reward for each arm. Each node RP_j generates $(\mathsf{pk}_{\mathsf{RP}_j}, \mathsf{sk}_{\mathsf{RP}_j})$ and shares $\mathsf{pk}_{\mathsf{RP}_j}$ with BAI and DO.

The algorithm, which is summarized in Algorithm 2, consists of:

– Initialization done by BAI is:
  - Based on the total budget N, compute $n_1, \dots, n_{K-1}$ that is the number of times each of the candidate arms should be pulled at phase $1, \dots, K-1$, respectively.
  - Uniformly select a permutation σ_1 of $[\![K]\!]$ and send $\mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_1}}(\sigma_1)$ to RP_1. A new permutation σ_j on $[\![K+1-j]\!]$ is randomly selected at each round, and sent to RP_j.
  - For each arm i, compute $sum[\sigma_1(i)] = \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(0)$.

  During the $K-1$ phases of the algorithm, these encrypted sums are updated by the nodes RP_j.
– $K-1$ phases where nodes BAI, RP_j, and Comp interact as shown in Fig. 2. We add the following specifications:
  - Each RP_j updates the encrypted sums using the homomorphic addition property of the Paillier's cryptosystem: for a round j and a candidate arm i with sum $sum[\sigma_j(i)]$, we get the updated sum $sum'[\sigma_j(i)]$ by homomorphically adding $\left(\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(x(\sigma_j(i)))\right)^{n_j} \times \prod_{l=1}^{n_j} \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(k_l)$ to $sum[\sigma_j(i)]$, where k_l is uniformly selected in $[-\epsilon, \epsilon]$ by RP_j.

- When Comp computes the index of the worst arm, if two or more arms have the same worst sum of rewards, then Comp selects uniformly at random one of these arms as the worst one. This ensures that the index of the worst arm has a uniform distribution.

Algorithm 2. Secure SR algorithms

```
function SETUP_BAI($N$) ▷ Step 1 ▷ $j$ tracks the round number, $sum$ contains the sum of rewards of each competing arm. Both are stored in BAI state.
    for j from 1 to K-1 do
        $n_j \leftarrow \left\lceil \frac{1}{\overline{log}(K)} \frac{N-K}{K+1-j} \right\rceil - \sum_{l=0}^{j-1} n_l$
    for all $i \in [\![K]\!]$ do
        $sum[i] \leftarrow \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(0)$
    $j \leftarrow 1$

function START_ROUND_BAI                                              ▷ Step 2*
    $\sigma_j \leftarrow$ random permutation of $[\![K - j + 1]\!]$
    save $\sigma_j$ in BAI state
    return $\sigma_j(sum), \mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j), n_j$

function ROUND_RP$_j$($\sigma_j(sum), \mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j), n_j$)                   ▷ Step 3*
    Decrypt $\mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j)$, retrieve $\sigma_j$ and un-permute $\sigma_j(sum)$ to get sum
    for each arm in sum do
        Homomorphically add to sum[arm] the rewards from $n_j$ pulls of the arm
    return $\sigma_j(sum)$

function ROUND_Comp($\sigma_j(sum)$)                                  ▷ Step 4*
    Decrypt each element of $\sigma_j(sum)$
    $x_{min} \leftarrow$ the index of a lowest element of the decrypted list, randomly chosen amongst all lowest elements
    return $x_{min}$

function END_ROUND_BAI($\sigma_j(sum), x_{min}$)      ▷ After Step 4*, before next round
    $u_{min} \leftarrow \sigma_j^{-1}(x_{min})$
    Remove arm $u_{min}$ from the list of participants, and from $sum$ to reflect so
    $j++$

function RESULT                                                        ▷ Step 5
    Get the only remaining competing arm $\widehat{i^*}$ from $sum$ in BAI state
    return $\mathcal{E}_{\mathsf{pk}_{\mathsf{U}}}(\widehat{i^*})$
```

We summarize in Table 1 what each cloud node knows and does not know, in order to satisfy the desired security properties. We formally prove the security properties in Appendix A. Next, we briefly outline why we need so many nodes:

- Assuming that all RP_j nodes are a single one, this node would know all rewards since the beginning of the algorithm hence it would learn the ranking of the arms.
- Assuming that Comp and RP_j are the same, then it would leak which arm is associated to which sum, hence the best arm could be leaked.
- Assuming that Comp and BAI are the same, then BAI would learn the plain rewards in addition to the ranking that it already knows.
- Assuming that BAI and RP_j are the same, then it would leak to BAI the sum of rewards associated to each arm.

3.4 Complexity

We give here a brief description of the complexity, in terms of the number of calls to $\mathcal{E}$ and $\mathcal{D}$ (the costliest operations).

- At Step 0, DO computes $\forall i \in [\![K]\!], \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(x(i))$. It also encrypts ϵ for each RP_j, thus having $O(K)$ complexity.
- At Step 2, BAI computes a new encrypted permutation, that can be encoded as $[\mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j(1)), \ldots, \mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j(K+1-j)]$, thus having $O(K-j) = O(K)$ complexity.
- At Step 3, RP_j computes the added rewards. Given the algorithm in Sect. 3.3, this step has $O(K)$ complexity.
- At Step 4, Comp decrypts all partial sums, with a complexity of $O(K)$, before sending the argmin to BAI.

Steps 2, 3, 4 are repeated $K-1$ times. The total complexity of these three steps is then $O(K^2)$, and the total complexity of the algorithm is $O(K^2)$.

Note that the complexity of the algorithm is independent from the total budget N, which is a great advantage as typical budgets for these kinds of problems are often elevated and usually much larger than the number of arms. More precisely, the complexity related to N is hidden by the complexity of the encryptions.

4 Experiments

We report on a proof-of-concept experimental study of our proposed protocol. We implemented and compared:

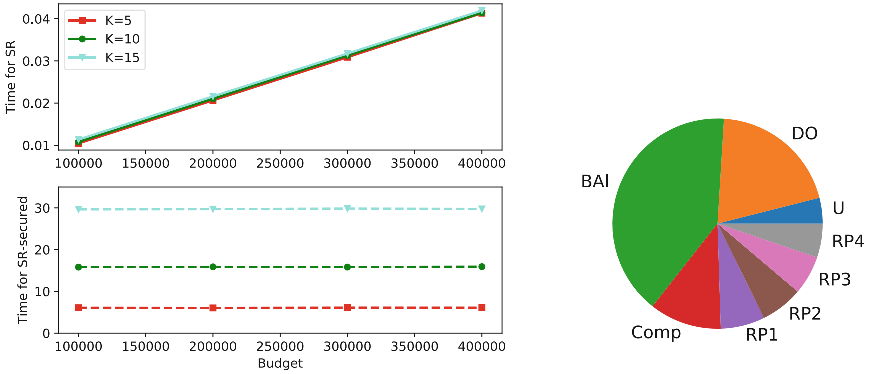
- SR: the successive rejects algorithm, adapted from [1]. We give the pseudocode of SR in Fig. 1.
- SR-secured: our proposed protocol, which adds security guarantees to SR. We describe SR-secured in Sect. 3.3 and we outline its workflow in Fig. 2.

We implemented the algorithms in Python 3. Our code is available on a public git repository[2]. For SR-secured, we used *phe*[3], an open-source Python 3 library for partially homomorphic encryption using the Paillier's cryptosystem.

[2] https://gitlab-sds.insa-cvl.fr/vciucanu/secure-bai-in-mab-public-code.

[3] https://python-paillier.readthedocs.io/en/develop/.

We summarize the results in Fig. 3 and we discuss them next. We carried out these experiments on a laptop with Intel Core i5 3.10 GHz and 8 GB of RAM. We used 2048 bits keys. The results are averaged over 100 runs.



(a) Run time (in seconds) for SR (full lines), and for SR-secured (dashed).

(b) Time share during SR-secured execution, for N=400000 and K=5.

Fig. 3. Experimental results.

Run Time Comparison SR vs SR-Secured. In each half of Fig. 3(a), we have 12 points, corresponding to the pairwise combinations between 4 budget values N (100000, 200000, 300000, 400000) and 3 values for the number of arms K (5, 10, 15). We split the figure in two plots with different Y axis because the observed times are in the order of tens of milliseconds for SR and tens of seconds for SR-secured. For SR, we observe that the time varies more on N and less on K, which makes sense because the operations depending on N (i.e., picking random numbers in the rewards generation) are more expensive than the operations depending on K (i.e., additions and multiplications). On the other hand, for SR-secured, the slight run time increase depending on N is barely visible (hence the curves look rather constant) because of the three-orders-of-magnitude overhead that is a natural consequence of the high number of encryptions and decryptions performed by SR-secured. As explained in Sect. 3.3, each participant and encryption/decryption from SR-secured is useful for the protocol in order to guarantee the desired security properties. We stress that the time of SR-secured barely grows when increasing the budget N, which confirms the essential property that we outlined in the complexity discussion: the number of cryptographic primitives does not depend on N. Hence, we were easily able to run SR-secured for large budgets as we show in the figure. We conclude from this experiment that SR-secured retains the scalability of SR while adding an overhead (depending on K and not on N) due to the security primitives. Obviously, both algorithms compute exactly the same result i.e., the best arm.

Moreover, before running this time comparison study, we carefully checked that all intermediate sums and arm rankings are identical for SR and SR-secured, despite the encryptions and decryptions that the latter algorithm performs.

Zoom on SR-Secured. In Fig. 3(b) we highlight how the total time taken by SR-secured is split among the participants. We obtained this figure for $N = 400000$ and $K = 5$, hence there are 4 phases, thus 4 participants $\mathsf{RP}_1, \mathsf{RP}_2, \mathsf{RP}_3, \mathsf{RP}_4$ in addition to $\mathsf{Comp}, \mathsf{BAI}$, the data owner DO, and the user U. First, notice that the shares of U and DO of the total time are relatively small, which is a desired property. Indeed, we require the DO only to encrypt her knowledge of the arms before outsourcing such encrypted data to the cloud (step 0 in Fig. 2). This could be actually done only once at the beginning and then all runs of the best-arm identification algorithm can be done using the same encrypted data, regardless of the user that pays for such a run. Moreover, we require U to not do any computation effort other than decrypting the result of the best-arm identification algorithm that the cloud returns to her (step 5 in Fig. 2). Among the cloud participants, we observe that BAI takes the lion's share of the total running time. This is expected because the role of BAI is similar to a controller that interacts with all other cloud participants. In what concerns Comp and the RP_j, their shares are quite similar. We observed the same behavior regardless of the chosen N and K on which we zoom.

5 Related Work

To the best of our knowledge, our work is the first that relies on public-key encryption in order to add privacy guarantees to best-arm identification algorithms for multi-armed bandits.

There is a recent line of research on multi-armed bandits using differential privacy techniques [5,6], which are based on adding an amount of noise to the data to ensure that the removal or addition of a single data item does not affect the outcome of any data analysis. These works have either focused on strategies to obtain: (i) privacy-preserving input guarantees i.e., make the observed rewards unintelligible to an outside user [9], or (ii) privacy-preserving output guarantees i.e., protect the chosen actions and their associated rewards from revealing private information [13,18].

There are some fundamental differences between this line of work based on differential privacy and our work based on public-key encryption. First, the considered multi-armed bandit problems are different. Indeed, we focus on identifying the best arm, which is equivalent to minimizing the *simple regret*, that is the difference between the values associated to the arm that is actually the best and the best arm identified by the algorithm. On the other hand, the aforementioned works consider the *cumulative regret* minimization that roughly consists of minimizing the difference between the rewards observed after pulling N times the best arm and the rewards observed during the N pulls done by the algorithm.

A second difference is as follows. On the one hand, our secured algorithm based on *public-key encryption* is guaranteed to return exactly the same result as the (non-secured) SR algorithm [1] on which we rely as a building block in our protocol. On the other hand, the result of a *differentially-private* algorithm contains by definition some noise, hence it is different from the result of the algorithm without privacy guarantees.

Third, by construction, the performance measure (the regret) of our secure algorithm remains the same as for the non-secured version, since both versions use the same arm-pulling strategies (that is, the performed encryptions/decryptions have no influence on the choice of arms to be pulled). The price we pay for making the algorithm secure comes only in the form of additional time needed for the encryptions and decryptions. In contrast, in the differential privacy approach, noise is introduced in the inputs/outputs in order to guarantee that the algorithms are differentially private and this has a direct impact on the arm-selection strategies. Therefore, the performance of the differential private versions of the algorithms suffers an increased regret with respect to their non-secured versions by an additive [18] or a multiplicative factor [9,13].

6 Conclusions and Future Work

We studied the problem of best-arm identification in multi-armed bandits and we addressed the inherent privacy concerns that occur when outsourcing the data and computations to a public cloud. Our main contribution is a distributed protocol that computes the best arm while guaranteeing that (i) no cloud node can learn at the same time information about the rewards and about the ranking of the arms and (ii) by analyzing the messages communicated between the different cloud nodes, no information can be learned about the rewards or about the ranking. To do so, we relied on the partially homomorphic property of Paillier's cryptosystem. The overhead due to the security primitives of our protocol depends only on the number of arms K and not on the budget N. Our experiments confirmed this property.

Looking ahead to the future work, there are many directions for further investigation. For example, we plan to investigate whether we can leverage an addition-homomorphic cryptosystem other than Paillier's, which may be more efficient in practice and could help us reduce the run time gap between the secured and the non-secured algorithms that we observed in our proof-of-concept experimental study. Additionally, we plan to add privacy guarantees to other multi-armed bandit settings e.g., cumulative regret minimization [2] or best-arm identification in linear bandits [16], where the rewards of the arms depend linearly on some unknown parameter.

A Security Proofs

In this section, we prove that our secure algorithm presented in Sect. 3.3 satisfies the two desirable security properties outlined in Sect. 3.1: we prove the first property from Sects. A.2 to A.4, and the second property in Sect. A.5.

A.1 Notations and Security Hypothesis

For a node A, we note $data_{\mathsf{A}}$ the data to which A has access and $\mathcal{A}^{pb}(d)$ the answer of a Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$ having knowledge of d, trying to solve the problem pb. We recall that, in our notation conventions, $[\![K]\!]$ denotes the set of positive integers lower than or equal to K: $[\![K]\!] = \{1, \ldots, K\}$.

Lemma 1. *For a list $l = [l_1, \ldots, l_n]$, a permutation σ and the permuted list $\sigma(l) = [l_{\sigma(1)}, \ldots, l_{\sigma(n)}]$, a PPT adversary $\mathcal{A}(l_\sigma)$ cannot invert one element with probability better than random. More specifically, $P\left[\mathcal{A}^{\sigma^{-1}}(\sigma(l)) \in \{i, \sigma^{-1}(i)\}_{i \in [\![K]\!]}\right] = \frac{1}{K}$ where $\mathcal{A}$ returns a tuple $(i, g(i))$ and $g(i)$ is $\mathcal{A}$'s guess for the preimage of i.*

Proof. This is immediate, as all preimages are equally likely if σ is uniformly selected. □

Lemma 2. *Let $\mathcal{A}$ be a PPT adversary. Consider the adversarial game in which $\mathcal{A}$ choses three messages m_0, m_1, z and sends them to the challenger $\mathcal{C}$. $\mathcal{C}$ choses a random bit b, and returns a tuple (c_0, c_1, s) where $c_0 = \mathcal{E}_{\mathsf{pk}}(m_0), c_1 = \mathcal{E}_{\mathsf{pk}}(m_1)$, and $s = \mathcal{E}_{\mathsf{pk}}(m_b + z) = c_b \cdot \mathcal{E}_{\mathsf{pk}}(z)$. $\mathcal{A}$ must then guess the value of b.*

If $\mathcal{E}_{\mathsf{pk}}(\cdot)$ is IND-CPA secure, then $\mathcal{A}$ does not have any advantage in this adversarial game: $2\left|P\left[\mathcal{A}(c_0, c_1, s) = b\right] - \frac{1}{2}\right| < negl(\eta)$.

Proof. Assume there is a PPT adversary $\mathcal{O}$ able to win the game with significant advantage $x + \text{negl}(\eta)$: then $\mathcal{O}$ can guess b with probability $\frac{1}{2} + \frac{x}{2} + \text{negl}(\eta)$. We then prove that an PPT adversary $\mathcal{A}$ can break the IND-CPA property of Paillier. We can assume that when $\mathcal{O}$ is given c_0, c_0', s as input (where c_0' is another encryption of m_0), then the advantage of $\mathcal{O}$ is negligible: this gives us a lower bound of the advantage of $\mathcal{O}$ in a more general adversarial game.

Let us consider an IND-CPA game and an adversary $\mathcal{A}$, in which $\mathcal{A}$ choses m_0, m_1 and sends them to the challenger. The challenger randomly selects the bit b and sends back $c_b = \mathcal{E}_{\mathsf{pk}}(m_b)$. Then, $\mathcal{A}$ selects a message z and computes $\mathcal{E}_{\mathsf{pk}}(z)$, before computing $s = \mathcal{E}_{\mathsf{pk}}(m_b) \cdot \mathcal{E}_{\mathsf{pk}}(z)$. $\mathcal{A}$ also computes $c_0' = \mathcal{E}_{\mathsf{pk}}(m_0)$. Then, $\mathcal{A}$ calls $\mathcal{O}(c_0', c_b, s)$, retrieves (in polynomial time) from $\mathcal{O}$ the guessed value b^*, and returns b^*.

If $b = 0$, then $\mathcal{A}$ has actually called $\mathcal{O}(c_0', c_0, s)$, which guesses the correct b^* with probability $\frac{1}{2} + \text{negl}(\eta)$. On the other hand, if $b = 1$, then $\mathcal{A}$ has actually called $\mathcal{O}(c_0', c_1, s)$, which gives the correct b^* with probability $\frac{1}{2} + \frac{x}{2} + \text{negl}(\eta)$.

b being randomly chosen, $\mathcal{A}$ correctly guesses b with probability $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{x}{2}\right) + \text{negl}(\eta) = \frac{1}{2} + \frac{x}{4} + \text{negl}(\eta)$, thus yielding to $\mathcal{A}$ an advantage of $\frac{x}{2} + \text{negl}(\eta)$ in the IND-CPA game, in polynomial time. This is a contradiction with the fact that Paillier is IND-CPA secure. □

A.2 Security Proofs for BAI

Lemma 3. *From the data obtained at round j, a honest-but-curious* BAI *does not know the sum of the rewards of any arm. More precisely, for R the set of possible rewards, $\frac{|R|}{|R|-1}\left|P\left[\mathcal{A}^{reward}(data_{\mathsf{BAI}^j}) \in \{i, reward(i)\}_{i \in [\![K+1-j]\!]}\right] - \frac{1}{|R|}\right| <$*

negl(η) *where* $A^{reward}(data_{\mathsf{BAI}^j})$ *returns a tuple* $(i, g_{reward}(i)))$, *with* $g_{reward}(i)$ *being* $\mathcal{A}$*'s guess of the sum of rewards of* i.

Proof. At round j, BAI has access to the permuted list of the encrypted partial sums, as well as to the permutation σ_j and the index i_{min_j} of the lowest-ranking element from round j. From the first two arguments, BAI can access to the (unpermuted) list of the encrypted partial sums of the arms rewards $se_j = \left[\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\alpha_1}), \ldots,\ \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\alpha_{K+1-j}})\right]$, where the α_i are the arms still present in the algorithm at step j. So we can equivalently say that $data^j_{\mathsf{BAI}} = [se_j, i_{min_j}]$.

Assume that there exists a PPT adversary $\mathcal{O}$ able to break the above inequality, with advantage $x + \text{negl}(\eta)$: given $[se, i_{min_j}]$ as input, $\mathcal{O}$ returns some tuple (i, g_{reward}) where g_{reward} is the guessed reward of the arm α_i. The guess is correct with probability $\frac{1}{|R|} + \frac{|R|-1}{|R|}x + \text{negl}(\eta)$. Also note that, on average, $i = 1$ with probability $\frac{1}{K+1-j}$.

Let us consider a classical IND-CPA game as previously defined. When $\mathcal{A}$ receives $\mathcal{E}_{\mathsf{pk}}(m_b)$, they randomly chose $K - j$ cleartexts $r_1, \ldots, r_{K-j}$ and compute their ciphertexts $\mathcal{E}_{\mathsf{pk}}(r_1), \ldots, \mathcal{E}_{\mathsf{pk}}(r_{K-j})$. Then, $\mathcal{A}$ calls the oracle $\mathcal{O}(\mathcal{E}_{\mathsf{pk}}(m_b), \mathcal{E}_{\mathsf{pk}}(r_1), \ldots, \mathcal{E}_{\mathsf{pk}}(r_{K-j}))$ which returns (i, g_{reward}). If $i = 1$ and $g_{reward} \in \{m_0, m_1\}$ then $\mathcal{A}$ returns 0 or 1, respectively. Otherwise $\mathcal{A}$ returns a random guess.

Finally, $\mathcal{A}$ returns the good answer with probability $\frac{1}{K+1-j}\left(\frac{1}{|R|} + \frac{|R|-1}{|R|}x + \text{negl}(\eta)\right) + \left(1 - \frac{1}{K+1-j}\left(\frac{1}{|R|} + \frac{|R|-1}{|R|}x + \text{negl}(\eta)\right)\right)\frac{1}{2}$, i.e. with probability $\frac{1}{2} + \frac{1}{2}\frac{1}{K+1-j}\left(\frac{1}{|R|} + \frac{|R|-1}{|R|}x\right) + \text{negl}(\eta)$, which yields an advantage of $\frac{1}{K+1-j}\left(\frac{1}{|R|} + \frac{|R|-1}{|R|}x\right) + \text{negl}(\eta)$ to $\mathcal{A}$. Hence, $\mathcal{A}$ has a non-negligible advantage in the IND-CPA game, which is a contradiction with the fact that Paillier's cryptosystem is IND-CPA secure. □

Theorem 1. *From the data obtained up to round* j, *a honest-but-curious* BAI *does not know the sum of the rewards of any arm. More precisely, for* R *the set of possible rewards,* $\frac{|R|}{|R|-1} \times \left|P\left[A^{reward}(data_{\mathsf{BAI}^{\leq j}}) \in \{i, reward(i)\}_{i \in [\![K]\!]}\right] - \frac{1}{|R|}\right| <$ *negl*(η) *where the data* $data_{\mathsf{BAI}^{\leq j}}$ *is the data obtained by* BAI *during the first* j *rounds and* $reward(i)$ *is the reward of the i-th arm.*

Proof. We notice that $data_{\mathsf{BAI}^{\leq j}}$ is equal to $[data_{\mathsf{BAI}^1}, \ldots, data_{\mathsf{BAI}^j}] = [[se_1, i_{min_1}], \ldots, [se_j, i_{min_j}]]$. We know that each ciphertext from se_{j+1} results from the homomorphic addition of one ciphertext from se_j and one other unknown ciphertext[4]. Given Lemma 2, the set $[se_j, se_{j+1}]$ is indistinguishable from the set $[se_j, se'_{j+1}]$ where se'_{j+1} is a list of ciphertexts, unrelated to the ones in se_j. Hence, $data_{\mathsf{BAI}^{\leq j}}$ is indistinguishable from the list $[se_1, se'_2, \ldots, se'_j, i_{min_1}, \ldots, i_{min_j}]$, where se'_i is a list of ciphertexts unrelated to se'_j or se_1.

Assume that there exists a PPT adversary $\mathcal{O}$ able to break the above inequality, with an advantage of $x + \text{negl}(\eta)$. The data available to $\mathcal{A}$ basically consists of j iterations, of various sizes, of the problem addressed in Lemma 3. Then, if

[4] Namely, the ciphertext of the rewards of the arm i at round j.

$\mathcal{A}$ can solve our current adversarial game with non negligible advantage, $\mathcal{A}$ can immediately solve the problem in Lemma 3 with non negligible advantage (from one set of ciphertexts, $\mathcal{A}$ will generate other sets, and immediately places itself in the current problem). Because a non negligible advantage to the above problem breaks IND-CPA security, we conclude to a contradiction. □

A.3 Security Proofs for Comp

Lemma 4. *Let $j \in [\![K-1]\!]$. From the data received at the round j, a honest-but-curious* Comp *cannot infer the ranking of any arm. More specifically, $P\left[\mathcal{A}^{rank}(data_{\mathsf{Comp}^j}) \in \{i, ranking(i)\}_{i\in[\![K+1-j]\!]}\right] = \frac{1}{K+1-j}$.*

Proof. We have $data_{\mathsf{Comp}} = se_{\sigma_j} = [\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\sigma_j(\alpha_1)}), \ldots, \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\sigma_j(\alpha_{K+1-j})})]$, which can be decrypted by Comp to $s_{\sigma_j} = [sum_{\sigma_j(\alpha_1)}, \ldots, sum_{\sigma_j(\alpha_{K+1-j})}]$ where the α_i are the arms still present at round j. From this list of scores, Comp can infer the ranking of the permuted arms, i.e., compute the ranking any $A_{\sigma_j(i)}$ in polynomial time.

Assume there exists a PPT adversary $\mathcal{A}_{\text{rank}}$ capable of breaking the above equality. If $\mathcal{A}$ is able to predict the ranking of the arm i with advantage better than random, then $\mathcal{A}$ knows the ranking of A_i, namely $ranking(A_i)$. Knowing the ranking of all $A_{\sigma_j(i)}$, with probability better than random, $\mathcal{A}$ is then able to compute $\sigma_j^{-1}(i)$ with advantage better than random by identifying which $A_{\sigma_j(i)}$ matches A_i. Hence a contradiction with Lemma 1. □

Theorem 2. *Let $j \in [\![K-1]\!]$. From the data received until the round j, A honest-but-curious* Comp *cannot infer the ranking of any arm. More specifically, $P\left[\mathcal{A}^{rank}(data_{\mathsf{Comp}^{\leq j}}) \in \{i, ranking(i)\}_{i\in[\![K]\!]}\right] = \frac{1}{K+1-j}$.*

Proof. The proof is based on the proof of Lemma 4, with additional arguments similar to the ones of the proof of Theorem 1: because of Lemma 2, we can assume that we have j independent sets of unrelated permuted data. If an adversary $\mathcal{A}$ can break the above equality with non-negligible advantage in PPT, then we can construct an adversary who breaks the equality of Lemma 4 with non-negligible advantage, in PPT, which breaks Lemma 1, so we get a contradiction. □

Lemma 5. *Let $j \in [\![K-1]\!]$. From the data received at round j, a honest-but-curious* Comp *does not know the correspondence between sums of rewards and arms. More specifically, $P\left[\mathcal{A}^{rwd}(data_{\mathsf{Comp}^j}) \in \{i, reward(i)\}_{i\in[\![K+1-j]\!]}\right] = \frac{1}{K+1-j}$.*

Proof. Assume that Comp is able, from s_{σ_j}, to infer the sum of rewards of the arm A_k with probability better than $\frac{1}{K+1-j}$. Because Comp knows the sum of rewards of the permuted arms $A_{\sigma_j(\alpha_i)}$ for all $i \in [\![K+1-j]\!]$, then by matching these rewards with the sum of the rewards of A_i, Comp is able to compute $\sigma_j(i)$ with a probability better than random. Hence, Comp breaks Lemma 1. □

Theorem 3. *Let $j \in [\![K-1]\!]$. From the data received until round j, a honest-but-curious* Comp *does not know the correspondence between sums of rewards and arms. More specifically, $P\left[\mathcal{A}^{rwd}(data_{\mathsf{Comp}^{\leq j}}) \in \{i, reward(i)\}_{i\in[\![K]\!]}\right] = \frac{1}{K}$.*

Proof. Similar to the proof of Theorem 2. □

A.4 Security Proofs for the RP_j

Theorem 4. *A honest-but-curious* RP_j *does not know the ranking of the* $K-j+1$ *best ranking arms, for* $j \in [\![K-1]\!]$. *More specifically,* $\forall j \in [\![K-1]\!], \forall i \in [\![K+1-j]\!]$, *and* $ranking_j(i)$ *is the ranking of the i-th arm at round j,* $\frac{K+1-j}{K-j}\left|P\left[\mathcal{A}^{rank}(data_{\mathsf{RP}_j}) \in \{i, ranking_j(i)\}_{i\in[\![K+1-j]\!]}\right] - \frac{1}{K+1-j}\right| < negl(\eta)$.

Proof. We have $data_{\mathsf{RP}_j} = [se_{\sigma_j}, \mathcal{E}_{\mathsf{pk}_{\mathsf{RP}_j}}(\sigma_j), n_j]$, where $se_{\sigma_j} = \sigma_j([\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\alpha_1}), \ldots, \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\alpha_{K-j}})])$, the permuted list of encrypted sums of rewards. RP_j can further 'un-permute' se_{σ_j} to $se = [\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\alpha_1}), \ldots, \mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(sum_{\alpha_{K-j}})]$, the list of encrypted sums of rewards. Note that n_j does not carry any information about the partial sum, as one can simulate any se with the same n_j, so does not carry significant information to our problem.

Assume that RP_j can guess the ranking of one element with advantage $x + \text{negl}(\eta)$: there exist a PPT oracle $\mathcal{O}$ taking as input se, and outputs $(i, v(i))$, with $i \in [\![K+1-j]\!]$. Furthermore, we have $\widehat{v}(i) = ranking_j(i)$ with probability $\frac{1}{K+1-j} + \frac{K-j}{K+1-j}x + \text{negl}(\eta)$. Note that, on average, $i = 1$ with probability $\frac{1}{K+1-j}$. Let us consider an IND-CPA game, in which the strategy of $\mathcal{A}$ is the same as the one in the proof of Lemma 3 (i.e., generate enough ciphertexts so they can call $\mathcal{O}$). Then, following the same reasoning we get that $\mathcal{A}$ has an advantage of $\frac{1}{K+1-j}\left(\frac{1}{K+1-j} + \frac{K-j}{K+1-j}x\right) + \text{negl}(\eta)$ in the IND-CPA game, which is a contradiction with the IND-CPA property of Paillier's. □

Theorem 5. *A honest-but-curious* RP_j *does not know the ranking of the* $j-1$ *lowest ranking arms. More specifically,* $\forall j \in \{3, \ldots, K-1\}, \forall i \in [\![j-1]\!]$, *and* $ranking(i)$ *the ranking of the i-th arm,* $\frac{j-1}{j-2}\left|P\left[\mathcal{A}^{rank}(data_{\mathsf{RP}_j}) \in \{i, ranking(i)\}_{i\in[\![K+1-j]\!]}\right] - \frac{1}{j-1}\right| < negl(\eta)$.

Proof. This is straightforward as RP_j does not receive any information about the sums of the j lowest ranking arms. Furthermore, we must impose $j \geq 3$ because it is clear that RP_1 and RP_2 know the ranking of the lowest ranking arm. □

Theorem 6. *Except for* RP_1, *a honest-but-curious* RP_j *does not know the sums of rewards at step j. More precisely, for R the set of possible rewards,* $\forall i \in [\![K+1-j]\!]$, $\frac{|R|}{|R|-1}\left|P\left[\mathcal{A}^{reward}(data_{\mathsf{RP}_j}) \in \{i, reward_j(i)\}_{i\in[\![K+1-j]\!]}\right] - \frac{1}{|R|}\right| < negl(\eta)$.

Proof. Assume that a PPT adversary $\mathcal{A}$ breaks the above inequality: there exists a PPT oracle $\mathcal{O}(c_1, \ldots, c_K)$, that returns the tuple (i, m_i) where m_i is the cleartext of c_i with advantage $x + \text{negl}(\eta)$. Then we prove that the adversary breaks the IND-CPA property of Paillier's cryptosystem. Note that, on average, $i = 1$ with probability $\frac{1}{n}$, and that a decryption is correct with probability $\frac{1}{|R|} + \frac{|R|-1}{|R|}x + \text{negl}(\eta)$.

If we consider an IND-CPA game where the strategy of $\mathcal{A}$ is the same as in the proof of Lemma 3 (i.e., generate enough ciphertexts so they can call $\mathcal{O}$), we get that $\mathcal{A}$ has an advantage of $\frac{1}{n|R|} + \frac{|R|-1}{n|R|}x$ in the IND-CPA game, which is a contradiction with Paillier being IND-CPA secure. □

A.5 Security Proof for an External Observer

Theorem 7. *An external observer, having access to the set M of all the messages exchanged during the protocol, cannot infer anything about the sum of rewards of any arm. More specifically, any such observer is bound by the inequality mentioned in Theorem 1, with $data_{\mathsf{BAI}^{\leq j}}$ being replaced by M.*

Proof. Assume that there exists an adversary $\mathcal{O}$ able to break the above inequality, given M, in PPT. We then prove that an adversary $\mathcal{A}$ is able to break IND-CPA security of Pailier's scheme in PPT.

Let us consider a classical IND-CPA challenge, in which $\mathcal{A}$ choses two rewards r_0, r_1 and sends them to the challenge. The challenger returns $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(r_b)$, where b is a uniformly random bit. Then, $\mathcal{A}$ simulates a secure multi-armed bandit protocol, with 2 arms, so that at the end of round 1, one of the arms has for encrypted sum of rewards the value $\mathcal{E}_{\mathsf{pk}_{\mathsf{Comp}}}(r_b)$, the other being random. This is possible because in this simulation, $\mathcal{A}$ can set herself the rewards x_i of each arm, as well as the budget for round 1. Furthermore, knowing the cleartext of every encrypted value at any time, $\mathcal{A}$ can simulate the full protocol by herself (especially, she can simulate Comp execution). This simulation yields a set of messages M.

Now, calling $\mathcal{O}(M), \mathcal{A}$ will retrieve in PPT, with some non-negligible advantage, some information about the sums of rewards of one of the arms. With probability $\frac{1}{2}$, this information will be about the arms of r_b, thus giving, in PPT, a non-negligible advantage in the IND-CPA game, as $\mathcal{A}$ is able to find the value of b with some advantage. This is a contradiction with the fact that Paillier is IND-CPA secure. □

Theorem 8. *An external observer, having access to the set M of all the messages exchanged during the protocol, cannot infer anything about the ranking of any arm: $\frac{K}{K-1}\left|P\left[\mathcal{A}^{rwd}(M) \in \{i, ranking(i)\}_{i\in[\![K]\!]}\right] - \frac{1}{K}\right| < negl(\eta)$.*

Proof. It is obvious that such an observer can deduce the permuted list of rankings by listening to data exchanged at step 4. However, from the data of one round, it is impossible to know more: the data from one round is an encrypted permuted sum of rewards S, the lowest permuted index i, and the same sum, with the lowest element removed S' (steps 3, 4, 2). This is equivalent of having knowledge of S and i only. If an adversary $\mathcal{O}$ breaks the inequality with S and i, then we can break IND-CPA.

Let $\mathcal{A}$ be the adversary, picking $K+1$ messages such that $m_0 < m'_i < m_1$, and a permutation σ. Sending m_0 and m_1, they receive $c_b = \mathcal{E}_{\mathsf{pk}}(m_b)$, and also compute $c'_i = \mathcal{E}_{\mathsf{pk}}(m'_i)$. Then, if $\mathcal{O}(\sigma([c_b, c'_2, \ldots, c'_k]), \sigma(0)) = 0$, $\mathcal{A}$ returns 0, else 1. If $\mathcal{O}$ has a non-negligible advantage x, we prove similarly to the other proofs that $\mathcal{A}$ has a advantage of $\frac{x}{2}$ in the IND-CPA game, which is a contradiction.

Now, because of Lemma 2, having access to all messages does not change anything. This is because each new round is indistinguishable from a simulation run by $\mathcal{A}$, so an advantage in the "all-rounds" game would yield an advantage in the "one-round" game. □

References

1. Audibert, J., Bubeck, S., Munos, R.: Best arm identification in multi-armed bandits. In: Conference on Learning Theory (COLT) (2010)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**, 235–256 (2002)
3. Chen, S., Lin, T., King, I., Lyu, M.R., Chen, W.: Combinatorial pure exploration of multi-armed bandits. In: Conference on Neural Information Processing Systems (NIPS) (2014)
4. Coquelin, P., Munos, R.: Bandit algorithms for tree search. In: Conference on Uncertainty in Artificial Intelligence (UAI) (2007)
5. Dwork, C.: Differential privacy. In: International Colloquium on Automata, Languages and Programming (ICALP) (2006)
6. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**, 211–407 (2014)
7. Even-Dar, E., Mannor, S., Mansour, Y.: Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. J. Mach. Learn. Res. **7**, 1079–1105 (2006)
8. Gabillon, V., Ghavamzadeh, M., Lazaric, A.: Best arm identification: a unified approach to fixed budget and fixed confidence. In: Conference on Neural Information Processing Systems (NIPS) (2012)
9. Gajane, P., Urvoy, T., Kaufmann, E.: Corrupt bandits for preserving local privacy. In: Algorithmic Learning Theory (ALT) (2018)
10. Kaufmann, E., Cappé, O., Garivier, A.: On the complexity of best-arm identification in multi-armed bandit models. J. Mach. Learn. Res. **17**, 1–42 (2016)
11. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_29
12. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: International Conference on World Wide Web (WWW) (2010)
13. Mishra, N., Thakurta, A.: (Nearly) optimal differentially private stochastic multi-arm bandits. In: Conference on Uncertainty in Artificial Intelligence (UAI) (2015)
14. Munos, R.: From bandits to Monte-Carlo tree search: the optimistic principle applied to optimization and planning. Found. Trends Mach. Learn. **7**, 1–129 (2014)
15. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
16. Soare, M., Lazaric, A., Munos, R.: Best-arm identification in linear bandits. In: Conference on Neural Information Processing Systems (NIPS) (2014)
17. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**, 285–294 (1933)
18. Tossou, A.C.Y., Dimitrakakis, C.: Algorithms for differentially private multi-armed bandits. In: AAAI Conference on Artificial Intelligence (2016)