# Secure Protocols for Best Arm Identification Using Secret Sharing Schemes

Shanuja Sasi[1], Asaf Cohen[2] and Onur Günlü[1,3]
[1] Information Theory and Security Laboratory (ITSL), Linköping University, Sweden
[2] The School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel
[3] Lehrstuhl für Nachrichtentechnik, TU Dortmund, Germany
E-mail: {shanuja.sasi, onur.gunlu}@liu.se, coasaf@bgu.ac.il

***Abstract*—This paper addresses the challenge of best arm identification in stochastic multi-armed bandit (MAB) models under privacy-preserving constraints, such as in dynamic spectrum access networks where secondary users must privately detect underutilized channels. While previous network security research has explored securing MAB algorithms through techniques such as homomorphic encryption or differential privacy, these methods often suffer from high computational overhead or introduce noise that strictly decreases accuracy. In contrast, this work focuses on lightweight solutions that ensure data confidentiality without compromising the accuracy of best arm identification. We introduce two secure protocols that leverage additive secret sharing and threshold secret sharing. The proposed model, employing aggregation nodes and a comparator node, securely distributes computations to prevent any entity from accessing complete reward or ranking data. Furthermore, the protocol ensures resistance to collusion and fault tolerance, while maintaining computational efficiency. These contributions establish a scalable and robust framework for privacy-preserving best arm identification, offering practical and secure solutions that use MAB methods for network security.**

## I. Introduction

In the stochastic *multi-armed bandit (MAB)* model, an agent repeatedly chooses which arm to pull from a set of $K$ arms, each yielding a stochastic reward from a distribution associated with that arm. Initially, the agent has no information on the expected reward of each arm. After each pull, the observed reward guides the agent in refining its strategy for selecting future arms based on a defined optimization goal. This model applies broadly, including in clinical trials to identify optimal treatments, online advertising, recommendation systems, dynamic spectrum access networks, and gaming.

The objective of this study is *best arm identification* in MAB. Given $K$ arms and a budget of $N$ pulls, the agent aims to allocate pulls to maximize the chance of identifying the arm with the highest expected reward. This objective is well-studied in the machine learning field [1]–[4], including from a security viewpoint [5]–[8]. In secure MAB protocols, a key challenge is ensuring that an adversary cannot infer sensitive information from the agent's choices or observations. This concern is particularly relevant in dynamic spectrum access networks [9], where secondary users must detect and utilize underused communication channels while keeping their sensing data private. A robust protocol should provide privacy guarantees while still enabling accurate best arm identification. Securing bandit algorithms using cryptographic techniques aims to maintain the same decision-making performance as traditional methods while enhancing security. Prior works on secure bandits have addressed different aspects of this problem, including best arm identification, cumulative reward maximization, and simple regret minimization, where the goal is to reduce the gap between the true best arm and the one identified by the algorithm [5]–[8]. While these secure algorithms produce the same outputs as their non-secure counterparts, the added security comes at a cost: the use of cryptographic primitives increases computational overhead, making the process more time-consuming. Recent work also explores differential privacy in MAB algorithms [10]–[13] by adding noise to the data to safeguard privacy. Research primarily targets: privacy-preserving input, shielding rewards from external access and privacy-preserving output, preventing private information leakage in selected actions and rewards. However, this approach differs fundamentally from cryptographic methods. Cryptographic protocols yield identical results to standard algorithms, while differential privacy introduces noise, altering outcomes from the original algorithm.

In this paper, we introduce two secure protocols for best arm identification in a MAB problem to keep all data private. The proposed protocols offer information-theoretic security, ensuring absolute protection against, irrespective of computational power. Unlike cryptographic security used in the previous works, which relies on computational assumptions and may become vulnerable to advances like quantum computing, information-theoretic security guarantees perfect secrecy, making it crucial for highly confidential scenarios. The first protocol uses a technique called threshold secret sharing. This splits the data into several pieces so that no single piece reveals any information about the data. A number of these pieces must be combined to recover the original data. The second protocol uses additive secret sharing, which splits reward values into two parts and gives them to separate nodes to process securely. Both methods ensure that no single party has complete access to the data. The data client, who wants to find the best arm, does not do any calculations on its own. Instead, it hands over the work to special external nodes. These nodes handle tasks like collecting rewards and comparing options, but they do it in a way that keeps the data private. The data client only organizes the order of the arms and learns which arm is eliminated after each step, but the data client never see the actual reward values or the total sums. To ensure security, the paper introduces two types of nodes: *Aggregation Nodes (ANs)*, which securely collect the sums of rewards, and a *Comparator Node (CN)*, which facilitates the comparison of arms. These nodes are designed with strict access limitations, ensuring that no single node has access to both the reward sums and the rankings simultaneously, thereby safeguarding the confidentiality of the best arm's identity. The system is also designed to be resilient against collusion and operational failures. In the first protocol, sensitive data is fragmented such that a minimum threshold of pieces is required for reconstruction, preventing small groups of colluding nodes from gaining access to critical information. Moreover, the framework is fault-tolerant, i.e., capable of maintaining functionality even when a subset of nodes fails. Both protocols utilize lightweight techniques, namely additive and threshold secret sharing, avoiding computationally expensive operations like homomorphic encryption. This ensures their efficiency and suitability for real-world applications.

In summary, our protocols offer a robust, privacy-preserving, and computationally efficient approach to identifying the best arm in a MAB problem. By protecting sensitive data, mitigating the risk of collusion, and ensuring practical usability, these methods provide a network security solution, which we have validated through empirical evaluation.

## II. Best Arm Identification Problem and Successive Rejects Algorithm

In this section we provide a background on the best arm identification problem.

*Best Arm Identification (BAI)*: The algorithm takes the number of arms $K$ and the budget $N$. Each arm $i$ (where $i \in [1, K]$) has an unknown distribution characterized by a reward function pull$(i)$, which returns a reward drawn from the distribution associated with arm $i$, having a mean reward $m_i$. The goal is to identify the best arm $i^*$, which is defined as the arm with the highest mean reward: $i^* = \arg\max_{i \in [1,K]} m_i$. This framework accommodates various types of probability distributions. The budget $N$ signifies the total number of arm pulls (and implicit reward observations) permitted for the learner. The output is the estimated best arm $\hat{i}^*$ determined after $N$ arm pulls, based on an allocation strategy that dictates how the budget is distributed across the $K$ arms. To optimize the budget allocation, the learner only knows the number of arms $K$ and the total budget $N$, lacking any prior information regarding the reward distributions for each arm. Essentially, the learner relies solely on the outcomes from the pull$(i)$ function. The observed rewards should provide valuable information for accurately identifying $i^*$. A larger budget facilitates more pulls, thereby enhancing the likelihood of correctly identifying the best arm. The challenge lies in maximizing the probability of correctly identifying the best arm within the limited budget.

*Successive Rejects (SR)*: The SR algorithm [1], as described in **Algorithm 1**, takes the number of arms $K$ and budget $N$ as inputs. Initially, all $K$ arms $[1, K]$ are considered as candidates. The algorithm divides the budget into $K-1$ phases.

**Algorithm 1** Successive Rejects Algorithm

```
Require: Number of arms $K$, Budget $N$
 1: Initialize set of candidate arms $A_1 = \{1, 2, \ldots, K\}$
 2: $n_0 = 0$
 3: $\Lambda_K = \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$
 4: for phase $j = 1$ to $K - 1$ do
 5:     Pull each candidate arm in $A_j$ for
        $n_j = \left\lceil \frac{1}{\Lambda_K} \frac{N-K}{K+1-j} \right\rceil - \sum_{l=0}^{j-1} n_l$ times
 6:     Compute the sum of rewards $s_i$ for each arm $i \in A_j$
 7:     Reject arm $i_m = \arg\min_{i \in A_j} s_i$ (if tie, reject one randomly)
 8:     Set $A_{j+1} = A_j \setminus \{i_m\}$
 9: end for
10: return The best arm $i^* = A_K$
```

At the conclusion of each phase, a decision is made regarding which arm to reject. The lengths of the phases are determined to ensure that the budget is not exceeded while minimizing the probability of incorrectly identifying the best arm. In each phase $j$ (where $j \in [1, K-1]$), every candidate arm in $A_j$ is pulled $n_j$ times according to a predefined allocation, where $A_j$ represents the set of active candidate arms at phase $j$. At the end of each phase, the arm with the lowest total observed rewards is rejected, as it is estimated to be the least promising. In the event of a tie, the algorithm randomly selects one of the worst arms to discard. In the subsequent phase, the remaining arms are again pulled according to the fixed allocation, resulting in the worst arm being pulled $n_1$ times, the second worst pulled $n_2 + n_1$ times, and so on, culminating in the best arm and the second-best arm being pulled $n_{K-1} + \ldots + n_1$ times. The estimated best arm is the only arm remaining after phase $K-1$.

### A. Cryptographic Security for BAI

In [8], three secure protocols (*SR-Ring*, *SR-Centralized*, and *SR-Paillier*) for BAI in MAB settings are provided. These protocols address privacy concerns in federated learning scenarios by ensuring that data remains confidential.

*Security Model:* The protocols are based on cryptographic security guarantees, which relies on the difficulty of solving specific mathematical problems (e.g., factoring large numbers in RSA). This type of security is effective as long as adversaries cannot perform these computations within a practical time frame. This framework operates under an honest-but-curious model, where all participants, such as *Data Owners (DOs)*, servers, and orchestration nodes, strictly adhere to the protocol's computational steps but may passively inspect received messages. While participants cannot tamper with data or collude with others, the model ensures robust privacy guarantees: (1) No DO can infer the identity of the best arm identified by the protocol or access the cumulative reward values of other DOs; (2) Servers, even when coordinating computations, cannot learn the best arm or aggregated reward sums. Furthermore, external observers, despite having full access to all network communications, are unable to deduce sensitive information such as the best arm or reward distributions.

*SR-Ring*: SR-Ring is a secure protocol designed to identify the best arm using a distributed ring computation. It involves *Data Owners (DOs)*, a central orchestration server, and a *Data Customer (DC)*. Each DO computes and updates its reward sums locally, and the arm to reject is determined in a ring-like structure. The protocol employs AES-CBC (cipher block chaining) encryption, a symmetric cryptographic scheme, to secure communication between participants. A shared symmetric key ensures that only authorized parties can decrypt the exchanged messages. Additionally, random permutations are used to hide the indices of arms, preventing the server from inferring the rankings. While it ensures robust security, its distributed nature and repeated encryptions result in high computational overhead. It is well-suited for scenarios where only one orchestrator is available.

*SR-Centralized*: SR-Centralized enhances the BAI process by introducing a trusted participant, called the Comparator, alongside the central server. The Comparator decrypts and compares the encrypted sums of rewards to determine the arm to reject. The protocol uses AES-CBC encryption, with keys shared only between the Comparator and the DOs, ensuring the server cannot access raw sums of rewards. Like *SR-Ring*, it employs random permutations to anonymize arm rankings. This protocol is faster than *SR-Ring* because it centralizes the rejection decision with the Comparator. However, it requires two non-colluding orchestrators to ensure security.

*SR-Paillier*: SR-Paillier is specifically tailored for uniform reward distributions and minimizes communication overhead for DOs. Each DO encrypts its reward values using the Paillier cryptosystem, a public-key encryption scheme supporting additive homomorphism. The server computes sums of rewards directly in the encrypted domain using Paillier's homomorphic properties. The Comparator decrypts the sums and identifies the arm to reject. Although SR-Paillier significantly reduces communication costs, it introduces high computational overhead due to Paillier encryption and decryption operations. It is best suited for scenarios where DOs cannot engage in frequent communication but can tolerate slower computation.

## III. Secret Sharing Schemes

A secret sharing scheme [14] is a method that divides a secret $s$ into multiple shares, distributed among participants in such a way that the secret can only be reconstructed if a minimum number of shares, known as the threshold, is available. This ensures that participants with fewer than the threshold number of shares cannot infer any information about the secret, safeguarding its confidentiality. There are two primary types of secret sharing schemes. The first is *additive secret sharing*, where the secret $s$ is split into $n$ random shares such that their sum modulo a specific field (e.g., $\mathbb{Z}_p$) equals $s$. Each participant receives one share, and the secret is reconstructed by summing all shares. This method is computationally simple and efficient but requires all shares to be available for recovery. The second is *$(m, N)$ threshold secret sharing*, which allows reconstruction of the secret with any $m$ shares out of $N$. It uses polynomial interpolation for secure distribution: a random polynomial $P(x)$ of degree $m-1$ encodes the secret in its coefficients, and each participant is given a unique share $P(x_i)$. The secret is recovered using any $m$ shares through interpolation methods such as Lagrange interpolation or Vandermonde matrix inversion.

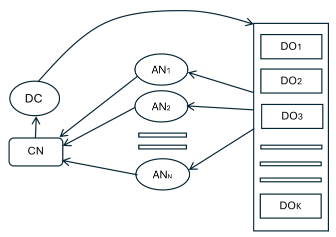

Fig. 1: Secure MAB Model.

## IV. Security Model for Secure MAB Protocols

The security model (as depicted in Fig. 1) and the relevant participants in this framework are provided below. The proposed secure protocols (Protocol 1 and 2) aims to maintain the same decision-making performance as the standard SR algorithm.

### A. Participants and Roles

- *Data Owners (DOs):* Each DO has private data associated with a specific arm. The DO pulls its arm and computes the sum of the rewards. Each DO has the total reward for its arm.
- *Data Client (DC):* The data client is the central coordinator seeking to identify the best arm. It manages the permutation of the arms during each phase of the process to protect the identities of the arms and the rewards associated with them. The DC does not have direct access to the rewards or the sum of rewards at any point. However, it does manage the ranking of arms during each round, and by the end of the process, it knows the order of arms but not the actual reward values.
- *Aggregator Nodes (ANs):* These nodes receive the secret shares of the reward sums from the DOs. Each aggregator node possesses only a piece of information, namely, a single share, ensuring that they cannot access the actual reward sums. The aggregator nodes are also unaware of the identified best arm.
  - Protocol 1: $N$ ANs ($\text{AN}_1$, $\text{AN}_2$, ..., $\text{AN}_N$), for some integer $N$. The number of aggregator nodes available is independent of the number of DOs and the number of pulls done at each round.
  - Protocol 2: 2 ANs (AN1 and AN2).
- *Comparator Node (CN):* The comparator node collects shares from the aggregator nodes to reconstruct the reward sums. It performs the comparisons and identifies the rejected arm while remaining unaware of the original identities of the arms.

- Arm anonymity: CN sees the sums of rewards but cannot map them to original arm indices.

### B. Information-theoretic Security Guarantees and Corruption Thresholds

In the proposed protocols, we provide information-theoretic security guarantees, which implies absolute protection against any adversary, regardless of their computational capabilities. Its security is grounded in fundamental mathematical principles. It achieves perfect secrecy which ensures that no amount of computational power can compromise the security. Protocol 1 utilizes $(m, N)$-threshold secret sharing, dividing data into $N$ shares, with at least $m$ shares required for reconstruction. This mechanism ensures strong resistance to collusion and offers fault tolerance, as it can handle up to $N - m$ node failures without compromising results. Protocol 2 employs additive secret sharing, splitting each reward sum into two shares. While computationally simpler and more efficient, Protocol 2 assumes that the two aggregation nodes do not collude, making its security guarantees weaker compared to Protocol 1. For a secret $s$, two shares are generated: $s^1$ and $s^2$ such that

- Additive shares satisfy the independence constraints $I(s; s^{(1)}) = I(s; s^{(2)}) = 0$.
- Collusion breaks confidentiality: $H(s|s^{(1)}, s^{(2)}) = 0$.

This is in contrast to the cryptographic security guarantees provided in the protocols in [8], which can be vulnerable if computational power or algorithms advance significantly (e.g., through quantum computing). On the other hand, information-theoretic security guarantees protection without such assumptions, making it especially valuable for scenarios requiring the utmost confidentiality.

## V. Protocol 1: Threshold Shielding

This section introduces a secure protocol for BAI. The protocol leverages $(m, N)$-*threshold secret sharing*, avoiding reliance on traditional cryptographic techniques such as AES-CBC or Paillier encryption. By incorporating additional nodes for aggregation and comparison, we ensure data privacy while facilitating efficient computation.

**Theorem 1.** ***(Perfect Secrecy of Protocol 1).*** *If N shares $\{a_j : j \in [1, N]\}$ are generated for a secret $s$ using $(m, N)$-threshold secret sharing, then for any $S \subset \{1, ..., N\}$ with $|S| < m$, we have $H(s|\{a_j\}_{j \in S}) = H(s)$.*

Theorem 1 directly follows from Shamir's scheme [14].

### A. Protocol Workflow

1) *Initialization:*
   - DC generates a random permutation $p$ of the arm indices. The purpose of this permutation is to anonymize the arms during the ranking process. It sends the permuted arm indices to the DOs. DC who is interested in identifying the best arm, allocates a budget $N$ (the total number of arm pulls) and sends it to DOs.
2) *Threshold Secret Sharing:*
   *Objective:* Securely distribute the reward sums of active arms across ANs to prevent any single node from accessing sensitive data, while ensuring fault tolerance and collusion resistance.
   Below are the inputs available at this phase:
   - *Active Arms:* A set $A_j$ of $(K - j + 1)$ arms remaining in round $j$.
   - *Reward Sums:* For each active arm $i \in A_j$, a sum $s_i$ of rewards observed after $n_j$ pulls.
   - *Threshold Parameters:* An $(m, N)$-threshold secret sharing scheme, where $m$ is the minimum shares required to reconstruct the secret and $N$ is the total number of shares generated per secret.

   The steps involved in this process are as follows:
   a) *Block Division*: Divide the active arms in $A_j$ into $B = \lceil \frac{K-j+1}{m} \rceil$ blocks, denoted as $[1, B]$. Each block contains $m$ arms (except the last block, which may have fewer).
   b) *Secret Sharing per Block*: For each block $b \in [1, B]$, the DOs combinedly generates $N$ shares $\{a_j^b : j \in [1, N]\}$ from the sum of rewards in block $b$ using a $(m, N)$-threshold secret sharing scheme. For each block $b \in [1, B]$, the share $a_j^b$ is sent to AN$_j$, for $j \in [1, N]$.

   *Security Guarantees:*
   - Any subset of less than $m$ shares reveals no information about the sum of rewards in a block. Even if $m-1$ ANs collude, they cannot reconstruct the secret.

   *Output*: Distributed shares $\{a_j^b\}$ stored across $N$ ANs.
3) *Summation and Comparison Phase*:
   *Objective:* Reconstruct the reward sums from shares, compare them, and identify the worst-performing arm to reject. The inputs to the CN are the shares $\{a_j^b\}$ obtained from the ANs.
   - The Comparator Node (CN) collects $m$ shares for each block $b$. Use polynomial interpolation (e.g., Lagrange) to recover the reward sums for each block.

     **Theorem 2.** ***(Fault Tolerance).*** *Protocol 1 tolerates up to $N - m$ node failures while correctly identifying the best arm, as only $m$ shares are needed for the reconstruction of each block.*
   - The CN ranks the reconstructed sums. The permuted arm with the lowest total reward sum is identified as the worst-performing arm.
4) *Rejection:*
   - The DC applies the inverse permutation $p^{-1}$ to recover the true index of the arm to be rejected. The DC informs the corresponding DO to stop pulling the rejected arm.
5) *Iteration and Best Arm Identification:*
   - Steps 2 through 4 are repeated with the updated set of active arms: $A_{j+1} = A_j \setminus \{\text{rejected arm}\}$, until only one arm remains. In the final step, the CN sends the result to the DC, which determines the true best arm using $p^{-1}$.

TABLE I: Comparison of protocols based on key features.

| **Protocol** | **# Nodes** | **Comp. Overhead** | **Security Features** | **Fault Tolerance** | **Scalability** |
|---|---|---|---|---|---|
| SR-Ring [8] | $K+2$ | High (AES-CBC, distributed) Require multiple rounds of substitution, shifting, mixing and XOR operations depending on the key size | Cryptographic security guarantees, Secure with random permutations | Low (no redundancy) | Quadratic with $K$ |
| SR-Centralized [8] | $K+3$ | High (AES-CBC, centralized) (everything else same as SR-Ring) | Cryptographic security guarantees, Requires non-colluding orchestrators (Server + CN) | Low | Quadratic with $K$ |
| SR-Paillier [8] | $K+3$ | Very High (Paillier) | Cryptographic security guarantees, Strong encryption but high latency | Low | Quadratic with $K$ |
| Protocol 1 | $K+N+2$ | Moderate (polynomial-based) (depends on m and N) | Information-theoretic security guarantees, Threshold sharing, Resistant to $m-1$ colluding nodes | High ($N-m$ failures) | Linear with $K$ |
| Protocol 2 | $K+4$ | Low (additive sharing) | Information-theoretic security guarantees, Assumes non-colluding ANs | Low (requires both ANs) | Constant per arm |

## VI. Protocol 2: Additive Shielding

The second protocol relies on additive secret sharing to split reward sums into two parts such that neither reveals any information about the original value individually.

### A. *Protocol Workflow*

1) *Initialization:*
   - The DC generates a permutation $p$ of arm indices, assigns pull budget $N$, and informs the DOs.
2) *Additive Secret Sharing:*
   - For active arm $i \in A_j$, DO$_i$ computes the sum $s_i$ after pulling its arm $n_j$ times during round $j$. The data owner splits the sum $s_i$ into two shares, $s_i^{(1)}$ and $s_i^{(2)}$, such that: $s_i = s_i^{(1)} + s_i^{(2)}$. $s_i^{(1)}$ is sent to AN1, and $s_i^{(2)}$ is sent to AN2.
3) *Summation by CN:*
   - CN aggregates the shares from AN1 and AN2, reconstructs the total sum of rewards for each active arm, handles the comparison and ranking of the sums. Arms are evaluated in the permuted order, and the arm with the lowest aggregated reward is rejected in each phase. It sends the index of the arm with the lowest aggregated reward back to DC.
4) *Rejection and Iteration:*
   - The DC identifies the rejected arm using $p^{-1}$ and notifies the corresponding DO. Steps 2 through 4 are repeated until the best arm is identified.

## VII. Protocol-Specific Analysis

An overall comparison of the proposed protocols with the cryptographic schemes in [8] is presented in Table I. The cryptographic protocols, namely SR-Ring, SR-Centralized, and SR-Paillier, ensure privacy based on cryptographic assumptions. All three exhibit high or very high overhead and scale quadratically with the number of arms $K$, with limited fault tolerance. In contrast, the proposed protocols focus on lightweight, information-theoretic security. *Protocol 1* uses $(m, N)$-threshold secret sharing, generating $\mathcal{O}(NK)$ shares using polynomials of degree $(m-1)$. It offers strong security and fault tolerance capable of surviving up to $N-m$ AN failures, but incurs moderate overhead and requires coordination among data owners, which may not be feasible in decentralized settings. *Protocol 2* adopts additive secret sharing, producing only two shares per arm per round ($\mathcal{O}(K)$ in total). It is highly efficient, scalable, and suitable for real-time applications, requiring no coordination among data owners.

## VIII. Empirical Evaluation

We conduct an experimental study to evaluate our protocols, considering the following algorithms: *SR*, *SR-Ring* [8], *SR-Centralized* [8], Protocol 1, and Protocol 2. SR-Paillier, designed for uniform reward distributions, is excluded from the comparison due to its significantly higher computational delay compared to SR-Ring and SR-Centralized in [8].

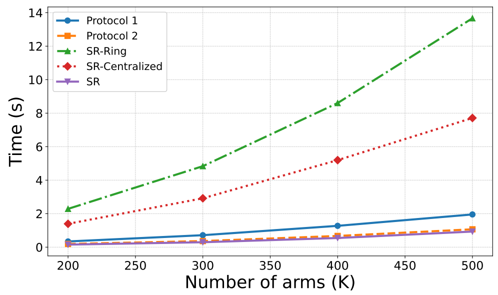

Fig. 2: Running time as the number of arms vary.

*1) Experiments on Synthetic Data:* To assess the performance and scalability of the proposed protocols, experiments are conducted on synthetic datasets. The evaluation focused on two types of distributions. For the *Bernoulli distribution*, rewards are assigned such that one arm have a reward of 0.8, another had 0.6, and the remaining arms have a reward of 0.7. For the *uniform distribution*, similar experiments are conducted, which exhibited comparable trends. The performance of the proposed protocols is compared against the cryptographic protocols SR-Centralized and SR-Ring.

*Scalability with Respect to the Number of Arms*: Protocol 1 demonstrates a linear increase in running time on the computer with the number of arms ($K$), as shown in Fig. 2. This scalability is attributed to its share generation mechanism, which utilizes polynomials of degree $m$. The number of required polynomials is determined by $\left\lceil \frac{K-j+1}{m} \right\rceil$, and each polynomial is evaluated at $N$ points to generate shares. For the experiments, key parameters were set to $N = 10$ (number of ANs) and $m = 6$, ensuring a balance between fault tolerance and computational cost. The assumption $N \ll K$ ensures efficiency in share generation. Despite the increasing number of arms, Protocol 1 consistently identified the best arm with minimal overhead due to its robust aggregation mechanism. Protocol 2 exhibits even greater scalability, showing minimal sensitivity to the number of arms. Unlike Protocol 1, where the number of shares depends on $N$, Protocol 2 generates exactly two shares per arm, regardless of $K$. This leads to a constant number of shares per arm, resulting in a significantly lower running time compared to Protocol 1. Furthermore, the aggregation and comparison processes in Protocol 2 are limited to simple summations, performed only once per phase. This design not only enhances runtime efficiency but also simplifies implementation, making it highly adaptable to various distribution types, including uniform distributions. The protocols SR-Centralized and SR-Ring both displayed quadratic growth in running time with the number of arms. This increase is a result of their reliance on cryptographic operations, making them less scalable than Protocol 1 and Protocol 2. In conclusion, Protocol 1 demonstrates a linear increase in running time with $K$, making it highly scalable for larger datasets. Protocol 2, on the other hand, maintains a consistent and minimal running time due to its fixed share generation process, demonstrating superior scalability. In contrast, SR-Centralized and SR-Ring show quadratic growth in running time with $K$, highlighting their limitations for larger datasets.

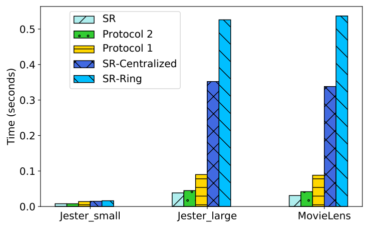

Fig. 3: Experimental results using Bernoulli reward: Running time on real world data scenarios.

*2) Experiments on Real-World Data:* We conduct experiments using real-world datasets to assess the performance and practicality of the proposed protocols. We use the same data and experimental setup as in [8]. The datasets utilized are *Jester*[1] and *MovieLens*[2], both of which include user ratings that are preprocessed into bandit scenarios. *Jester Dataset* includes ratings ranging from $-10$ (very not funny) to 10 (very funny), provided by 25,000 users for 100 jokes. Two scenarios are considered: *Jester-small* $K = 10$, corresponding to the 10

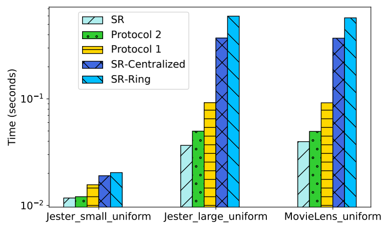

Fig. 4: Experimental results using uniform distribution: Running time on real world data scenarios.

most-rated jokes. The mean reward ($m_i$) for joke $i$ is computed as: $m_i = \frac{\text{\# of ratings } \geq 3.5 \text{ for joke } i}{\text{\# of users}}$ and *Jester-large* $K = 100$, corresponding to all jokes, with $m_i$ computed similarly but with a threshold of 7. *MovieLens Dataset* includes ratings ranging from 1 (bad) to 5 (very good), provided by 1,000 users for 100 movies. The following scenario is considered: $K = 100$, where the mean reward ($m_i$) for movie $i$ was computed as: $m_i = \frac{\text{\# of ratings } \geq 4 \text{ for movie } i}{\text{\# of users}}$. The results are presented in Fig. 3 and 4 and confirm the findings from synthetic data experiments.

## REFERENCES

[1] J. Audibert, S. Bubeck, and R. Munos, "Best arm identification in multi-armed bandits," in *Annu. Workshop Comput. Learn. Theory*, Jun. 2010, pp. 41–53.
[2] V. Gabillon, M. Ghavamzadeh, and A. Lazaric, "Best arm identification: A unified approach to fixed budget and fixed confidence," in *Proc. Neural Inf. Process. Syst.*, Lake Tahoe, Nevada, United States, Dec. 2012, pp. 3221–3229.
[3] E. Kaufmann, O. Cappée, and A. Garivier, "On the complexity of best-arm identification in multi-armed bandit models," *Jour. Mach. Learn. Research*, vol. 17, pp. 1–42, Jan. 2016.
[4] M. Soare, A. Lazaric, and R. Munos, "Best-arm identification in linear bandits," in *Proc. Neural Inf. Process. Syst.*, Montreal, Quebec, Canada, Dec. 2014, pp. 828–836.
[5] R. Ciucanu, P. Lafourcade, M. Lombard-Platet, and M. Soare, "Secure Best Arm Identification in Multi-Armed Bandits," in *Proc. Int. Conf. Inf. Security Pract. Experience*, Nov. 2019, pp. 152–171.
[6] R. Ciucanu, P. Lafourcade, M. Lombard-Platet, and M. Soare, "Secure outsourcing of multi-armed bandits," in *IEEE Int. Conf. Trust, Security Privacy Comput. Commun.*, Guangzhou, China, Dec. 2020, pp. 202–209.
[7] R. Ciucanu, A. Delabrouille, P. Lafourcade, and M. Soare, "Secure cumulative reward maximization in linear stochastic bandits," in *Proc. Int. Conf. Provable Security*, Nov. 2020, pp. 257–277.
[8] R. Ciucanu, A. Delabrouille, P. Lafourcade, and M. Soare, "Secure Protocols for Best Arm Identification in Federated Stochastic Multi-Armed Bandits," *IEEE Trans. Dependable Secure Computing*, vol. 20, no. 2, pp. 1380–1389, Apr. 2023.
[9] S. Kang and C. Joo, "Combinatorial multi-armed bandits in cognitive radio networks: A brief overview," *Int. Conf. Inf. and Commun. Tech. Convergence*, Jeju, Korea (South), Oct. 2017, pp. 1086-1088.
[10] P. Gajane, T. Urvoy, and E. Kaufmann, "Corrupt Bandits for Preserving Local Privacy," in *Alg. Learn. Theory*, Apr. 2018, pp. 387–412.
[11] R. Féraud, R. Alami, and R. Laroche, "Decentralized exploration in multi-armed bandits," in *Int. Conf. Mach. Learn.*, Long Beach, CA, USA, Jun. 2019, pp. 1901–1909.
[12] N. Mishra and A. Thakurta, "(Nearly) optimal differentially private stochastic multi-arm bandits," in *Conf. Uncertainty Artif. Intell.*, Amsterdam, Netherlands, Jul. 2015, pp. 592–601.
[13] A. C. Y. Tossou and C. Dimitrakakis, "Algorithms for Differentially Private Multi-Armed Bandits," in *AAAI Conf. Artif. Intell.*, Phoenix, Arizona, USA, Feb. 2016, pp. 2087–2093.
[14] A. Shamir, "How to share a secret," *ACM Commun.*, vol. 22, no. 11, pp. 612-613, Nov. 1979.

[1]http://eigentaste.berkeley.edu/dataset/
[2]hhttps://grouplens.org/datasets/movielens/