Appendix A. Pseudocode of the Extended Version

In Figure 9, we present the extended Samba pseudocode, which concerns the algorithms instantiated in Samba with an arbitrary number of iterations $nb_{\mathcal{A}}$ over the core of Samba. Figure 9 extends the pseudocode from Figure 6 and formalizes the intuitions given in Section 4.2 for **Pursuit**. For ease of readability, we highlight the differences with respect to Figure 6. The pseudocode of DC remains the same as in Figure 6.

```
/* Parameter setup */
receive N, A, seed_{α_t} from Controller                    (1)
nb_A ← number of iterations needed by A
r ← pull(i)
s_i ← r
n_i ← 1
/* Core of Samba */
for t ∈ [[K+1, N]]
  b_{i,t} = 0
  for l ∈ [[nb_A]]
    α_t ← nextValueFromSeed(seed_{α_t}, t, l)
    v_{i,t} ← ComputeScore(A, t, s_i, n_i, l, b_{i,t})
    send Enc(α_t v_{i,t}) to Controller                     (2)
    receive Enc(b_{i,t}) from Controller                    (5)
    b_{i,t} ← Dec(Enc(b_{i,t}))
  if b_{i,t} = 1
    r ← pull(i)
    s_i ← s_i + r
    n_i ← n_i + 1
/* Cumulative reward computation */
send E(s_i) to Controller                                   (6)
```

(a) Pseudocode of DO_i.

```
/* Parameter setup */
receive N, A from DC                                        (0)
send N, A to Comp                                           (1)
seed_{α_t} ← new random integer
for i ∈ [[K]]
  send N, A, seed_{α_t} to DO_i                             (1)
nb_A ← number of iterations needed by A
/* Core of Samba */
for t ∈ [[K+1, N]]
  for l ∈ [[nb_A]]
    for i ∈ [[K]]
      receive Enc(α_t · v_{i,t}) from DO_i                  (2)
    σ_t ← new random permutation
    send σ_t([Enc(α_t · v_{i,t})]_{i∈[[K]]}) to Comp        (3)
    receive σ_t([Enc(b_{i,t})]_{i∈[[K]]}]) from Comp        (4)
    for i ∈ [[K]]
      send σ_t^{-1}(σ_t(Enc(b_{i,t})) to DO_i               (5)
/* Cumulative reward computation */
for i ∈ [[K]]
  receive E(s_i) from DO_i                                  (6)
send E(s_1) · ... · E(s_K) to DC                            (7)
```

(b) Pseudocode of Controller.

```
/* Parameter setup */
receive N, A from Controller                                (1)
nb_A ← number of iterations needed by A
/* Core of Samba */
for t ∈ [[K+1, N]]
  for l ∈ [[nb_A]]
    receive σ_t([Enc(α_t · v_{i,t})]_{i∈[[K]]}) from Controller (3)
    V ← σ_t([Dec(Enc(α_t · v_{i,t}))]_{i∈[[K]]})
    σ_t(i_m) ← SelectArm(A, t, V)
    send σ_t([Enc(1_{i=σ_t(i_m)})]_{i∈[[K]]}) to Controller  (4)
```

(c) Pseudocode of Comp.

Figure 9: Extension of the Samba pseudocode cf. Figure 6 to instantiate bandit algorithms that require an arbitrary number of iterations $nb_{\mathcal{A}}$ over the core of Samba.

Appendix B. Omitted Proofs

We first prove the genericity theorem from Section 5.1.

Theorem 1 *A standard cumulative reward maximization algorithm $\mathcal{A}$ can* be instantiated in SAMBA *if the computations of $\mathcal{A}$ can be distributed using $nb_{\mathcal{A}}$ iterations over the core of* SAMBA *such that the following two properties hold:*

1. Arm score locality*: At each time step $t \in [\![K+1, N]\!]$ and iteration $l \in [\![nb_{\mathcal{A}}]\!]$, DO_i can evaluate the function* ComputeScore *based only on its local variables n_i and s_i.*
2. Oblivious arm selection*: At each time step $t \in [\![K+1, N]\!]$ and iteration $l \in [\![nb_{\mathcal{A}}]\!]$,* Comp *can evaluate the function* SelectArm *based only on the current list $\mathcal{V}$ of permuted masked arm scores.*

Proof. We prove by recurrence on the number of iterations $nb_{\mathcal{A}}$. First, we discuss the *initialization case*, which occurs when $nb_{\mathcal{A}} = 1$ i.e., only one iteration is done over the core of SAMBA. By assumption, $\mathcal{A}$ respects the two aforementioned *Arm score locality* and *Oblivious arm selection* stated in the theorem. At time step $t \in [\![K+1, N]\!]$, at Step 2, each DO_i computes his score depending only on its local s_i and n_i, as stated in the *Arm score locality* property. Once computed, the score is masked, encrypted and sent to Controller, which at Step 3, sends a permuted list of encrypted masked scores to Comp. Thanks to the *Oblivious arm selection*, Comp is able to apply the arm selection function on the permuted list of decrypted masked scores, in order to send back, at Step 4, the permuted list of encrypted pulling bits to Controller. Finally, at Step 5, Controller inverts the permutation of the received list and forwards each encrypted pulling bit to its associated DO_i. Thus, for a single iteration, by assuming that the federated version of algorithm $\mathcal{A}$ respects *Arm score locality* and *Oblivious arm selection*, SAMBA can be used to instantiate $\mathcal{A}$.

Next, we discuss the *inheritance on $nb_{\mathcal{A}}$*. We suppose that the theorem holds when there are $nb_{\mathcal{A}}$ iterations i.e., the federated version of $\mathcal{A}$ iterates over the core of SAMBA $nb_{\mathcal{A}}$ times, while respecting *Arm score locality* and *Oblivious arm selection*. We prove that the theorem still holds for $nb_{\mathcal{A}} + 1$ iterations over the core of SAMBA. By construction of SAMBA, the first $nb_{\mathcal{A}}$ iterations do not pull any arm, meaning that local variables n_i and s_i are not modified. At each iteration $l \in [\![nb_{\mathcal{A}}]\!]$, thanks to *Arm score locality* property, each DO_i computes its score locally from n_i and s_i, and thanks to *Oblivious arm selection*, Comp is able to select an arm based on the permuted list of masked scores, specific to algorithm $\mathcal{A}$. By adding a new iteration, both *ComputeScore* and *SelectArm* are called once more. By hypothesis, *ComputeScore* and *SelectArm* still respect *Arm score locality* and *Oblivious arm selection*, respectively. At iteration $l + 1$, at Step 2, each DO_i locally computes its score depending on s_i and n_i. Once computed, the score is masked, encrypted and sent to Controller, which at Step 3, sends a permuted list of encrypted masked score to Comp. Thanks to *Oblivious arm selection*, Comp is able to perform the arm selection on the permuted list of decrypted masked scores, in order to send back, at Step 4, the permuted list of encrypted pulling bits to Controller. Finally, at Step 5, Controller inverts the permutation of the received list and forwards each encrypted pulling bit to each DO_i. Thus, by assuming that the *ComputeScore* respects the *Arm score locality* property and that *SelectArm* respects the *Oblivious arm selection* property, adding a new iteration over the SAMBA's core allows SAMBA to instantiate $\mathcal{A}$. By recurrence, our theorem is proved. □

Next, we prove the security theorem from Section 5.3.

Theorem 3 *Take a standard cumulative reward maximization algorithm $\mathcal{A}$ and its federated version $\mathcal{A}'$ instantiated in* Samba *(cf. Theorem 1). It holds that $\mathcal{A}'$ satisfies the security properties summarized in Figure 7 if the primitives used in* Samba *are secure.*

For DO_i and DC, the arguments given in Section 5.3 suffice to show their security properties from Figure 7. We prove the security of Comp in Section B.1, $\mathsf{Controller}$ in Section B.2, and an external observer in Section B.3. We first state an useful lemma.

Lemma 1. *Take a list L and a random permutation $\sigma : L \to L$. Take an adversary Adv, who knows L, but does not know σ.*

- *Given $l \in L$, Adv cannot guess $\sigma(l)$ with probability better than random i.e., $\frac{1}{|L|}$.*
- *Given $\sigma(l) \in L$, Adv cannot guess l with probability better than random i.e., $\frac{1}{|L|}$.*

Proof. This is immediate because σ is randomly generated. □

B.1 Security of Comp

By construction of Samba, Comp computes the next arm to be pulled based on $\mathcal{V}$, which is the list of masked arm scores, permuted with σ_t (unknown to Comp). Because of Lemma 1, we infer that Comp cannot guess with probability better than random which arm is pulled at some time step t. Moreover, the permutation σ_t changes at each iteration, hence Comp does not know if there are some arms pulled more often than others.

As for the local variables of some DO_i (sum of rewards s_i and number of pulls n_i), we stress that Comp cannot guess such data with probability better than random. Indeed, although Comp sees arm scores in clear (that may be strongly correlated to s_i and n_i, depending on the algorithm), the real arm scores are hidden with a random multiplicative mask α_t that is unknown to Comp. Moreover, the mask α_t changes at each iteration, hence Comp cannot associate the scores from successive iterations in order to guess with probability better than random how s_i and n_i evolve from an iteration to the next one.

B.2 Security of Controller

We recall the formal statement that $\mathsf{Controller}$ cannot learn some sum of rewards with probability better than random, that we briefly introduced in Section 5.3.

Theorem 5. *For an arm $i \in [\![K]\!]$ and a time step $t \in [\![K+1, N]\!]$, an honest-but-curious* $\mathsf{Controller}$ *cannot guess $s_{i,t}$, given $data^t_{\mathsf{Controller}}$, with a probability better than random. More precisely, for all PPT adversaries* Adv*,*

$$\left| \Pr\left[(i, \hat{s}_{i,t}) \leftarrow \mathsf{Adv}^{sum(t)}(data^t_{\mathsf{Controller}}); \hat{s}_{i,t} = s_{i,t} \right] - p_S(s_{i,t}) \right|$$

is negligible in λ, where $\mathsf{Adv}^{sum(t)}(data^t_{\mathsf{Controller}})$ returns $(i, \hat{s}_{i,t})$ in which $\hat{s}_{i,t}$ is Adv's guess on $s_{i,t}$ for the arm i (chosen by Adv), and $p_S(s_{i,t})$ is the probability of obtaining a sum of rewards $s_{i,t}$ for an arm i at time step t.

Proof. $\mathsf{Controller}$ does not see in clear any data pertaining to the sums of rewards produced by DO_i. In particular, at Step 6, $\mathsf{Controller}$ receives $\mathcal{E}_{\mathsf{DC}}(s_{1,t}), \ldots, \mathcal{E}_{\mathsf{DC}}(s_{K,t})$. We prove that

retrieving $s_{i,t}$ from these ciphertexts breaks the IND-CPA property of Paillier's cryptosystem (Paillier, 1999). Toward a contradiction, we assume that there exists a PPT adversary Adv able, from $data^t_{\mathsf{Controller}}$ to find $s_{i,t}$ for some i with non negligible advantage x:

$$\left|\Pr\left[(i,\hat{s}_{i,t}) \leftarrow \mathsf{Adv}^{sum(t)}(data^t_{\mathsf{Controller}}); \hat{s}_{i,t} = s_{i,t}\right] - p_S(s_{i,t})\right| = x + \mathrm{negl}(\lambda).$$

Each $i \in [\![K]\!]$ has an equal probability of being chosen by Adv. We also assume that if $data^t_{\mathsf{Controller}}$ does not correspond to correct data collected by $\mathsf{Controller}$ during a run of SAMBA (for instance, if one piece of $data^t_{\mathsf{Controller}}$ has been replaced by another unrelated message), then Adv does not give any advantage. If such an adversary Adv exists, then we show how to construct an adversary B able to break the IND-CPA property of Paillier. We build an IND-CPA game, in which B chooses two values m_0, m_1, and sends them to the challenger. The challenger randomly selects $b \in \{0,1\}$ and answers with $\mathcal{E}_{\mathsf{DC}}(m_b)$. B wins the IND-CPA game if B guesses b with a non-negligible advantage. To do so, B first creates a simulation of a SAMBA execution i.e., B creates SAMBA participants, among which DO_i with Bernoulli distributions defined by μ'_i of its choice. Then, B runs an execution of SAMBA on these nodes. Because B controls all nodes, it knows the sums of rewards $s'_{1,t}, \ldots, s'_{K,t}$. As input for the IND-CPA game, B chooses $m_1 = s'_{1,t}$ and another value m_0, different from all $s'_{i,t}$, sends both values to the challenger, and receives $\mathcal{E}_{\mathsf{DC}}(m_b)$. Then, B computes $\mathcal{E}_{\mathsf{DC}}(s'_{i,t})$ for each i, and calls $\mathsf{Adv}^{sum(t)}([\mathcal{E}_{\mathsf{DC}}(m_b), \mathcal{E}_{\mathsf{DC}}(s'_{2,t}), \ldots, \mathcal{E}_{\mathsf{DC}}(s'_{K,t})])$. The strategy of B is as follows: if Adv returns $(1, m_1)$, then B answers 1. Otherwise, B answers randomly. We next derive the probability of a correct answer by B.

- If $i \neq 1$ (probability $1 - \frac{1}{K}$), then B answers randomly and is correct with probability $\frac{1}{2}$. Hence this branch offers a probability of success of $(1 - \frac{1}{K})\frac{1}{2}$.
- If $i = 1$ (probability $\frac{1}{K}$), we consider the value of b:
  - If $b = 0$ (probability $\frac{1}{2}$), then we have two cases:
    - If Adv outputs $(1, m_1)$ (probability $p_S(s_{1,t})$), then B answers 1 and it is wrong, hence the probability of success is 0.
    - Otherwise (probability $1 - p_S(s_{1,t})$), B answers randomly and is correct with probability $\frac{1}{2}$. This branch's success probability is $\frac{1}{K}\frac{1}{2}(1 - p_S(s_{1,t}))\frac{1}{2}$.
  - If $b = 1$ (probability $\frac{1}{2}$), then we have two cases:
    - If Adv outputs $(1, m_1)$ (probability $p_S(s_{1,t}) + x + \mathrm{negl}(\lambda)$), then B correctly answers 1. This branch's success probability is $\frac{1}{K}\frac{1}{2}(p_S(s_{1,t}) + x + \mathrm{negl}(\lambda))$.
    - Otherwise (probability $1 - p_S(s_{1,t}) - x - \mathrm{negl}(\lambda)$), B answers randomly and is correct with probability $\frac{1}{2}$. This branch's success probability is $\frac{1}{K}\frac{1}{2}(1 - p_S(s_{1,t}) - x - \mathrm{negl}(\lambda))\frac{1}{2}$.

By aggregating the aforementioned cases, the probability α of success of B is:

$$\begin{aligned}
\alpha =& \left(1 - \frac{1}{K}\right)\frac{1}{2} + \frac{1}{K}\frac{1}{2}\left(1 - p_S(s_{1,t})\right)\frac{1}{2} + \frac{1}{K}\frac{1}{2}\left(p_S(s_{1,t}) + x + \mathrm{negl}(\lambda)\right) \\
&+ \frac{1}{K}\frac{1}{2}\left(1 - p_S(s_{1,t}) - x - \mathrm{negl}(\lambda)\right)\frac{1}{2} \\
=& \frac{1}{2} - \frac{1}{2K} + \frac{1}{4K} - \frac{p_S(s_{1,t})}{4K} + \frac{p_S(s_{1,t})}{2K} + \frac{x}{2K} + \frac{1}{4K} - \frac{p_S(s_{1,t})}{4K} - \frac{x}{4K} + \mathrm{negl}(\lambda) \\
=& \frac{1}{2} + \frac{x}{4K} + \mathrm{negl}(\lambda)
\end{aligned}$$

Hence, B has an advantage of $\frac{x}{4K}$ in the IND-CPA game, which is non negligible. This is a contradiction with the fact that Paillier is IND-CPA secure. Consequently, there does not exist any PPT adversary Adv that violates the property stated in the theorem.

Moreover, to prove that Controller does not retrieve any piece of $s_{i,t}$ from encrypted scores it receives at Step 2, a similar reduction can be easily done by replacing the Paillier's cryptosystem with AES-GCM. □

We next state an useful lemma, which intuitively says that guessing the cumulative reward with probability better than random is equivalent to guessing the sum of rewards of some arm with probability better than random.

Lemma 2. *Let* Adv *be a PPT adversary trying to find the cumulative reward R, and let* B *be a PPT adversary trying to find the sum of rewards of some arm. Let d be some data, $cr(.)$ be the problem of guessing the cumulative reward, and $sum(.)$ be the problem of guessing the sum of rewards of some arm. We have the following statement:* $\mathsf{Adv}^{cr(.)}(d)$ *has a non-negligible advantage* $\Leftrightarrow$ $\mathsf{B}^{sum(.)}(d)$ *has a non-negligible advantage.*

Proof. $\Leftarrow$ Assume that B can guess the sum of rewards of some arm with probability better than random. Then, Adv can call B, and hence get the sum of rewards of one arm with probability better than random. From this sum, Adv can guess a lower bound on the cumulative reward, hence eliminating some possibilities, and thus guessing the cumulative reward with probability better than random.

$\Rightarrow$ If Adv can guess the cumulative reward with probability better than random, then B can use this cumulative reward as an upper bound on the sum of rewards of some arm, thus having a probability better than random of guessing the sum of rewards of some arm. □

As a corollary, by Lemma 2 and Theorem 5, we infer that Controller cannot guess the cumulative reward with probability better than random. We next present a result similar to Theorem 5, but concerning the arm that is pulled at some time step.

Theorem 6. *For each time step $t \in [\![K+1, N]\!]$,* Controller *cannot guess which arm is pulled at time step t with probability better than random.*

Proof. By construction of Samba, at time step $t \in [\![K+1, N]\!]$, Controller receives at Step 4 the permuted list of pulling bits, where each bit has been encrypted with AES-GCM with the symmetric key known only by Comp and all DO_i. We next show that if there exists a PPT adversary with a non negligible advantage in guessing the arm pulled at some time step $K+1 \leq t \leq N$, then this would break the IND-CPA property of AES-GCM. We denote by $data^t_{\mathsf{Controller}}$ this collection of data known by Controller after time step t. We assume, toward a contradiction, that there exists a PPT adversary Adv able from $data^t_{\mathsf{Controller}}$ to find the arm i_m pulled at some time step t with a non negligible advantage x:

$$\left| \Pr[\mathsf{Adv}^{pa(t)}(data^t_{\mathsf{Controller}}) = i^t_m] - \frac{1}{K} \right| = x + \mathrm{negl}(\lambda),$$

where $\mathsf{Adv}^{pa(t)}(data^t_{\mathsf{Controller}})$ returns the guess of Adv on which arm is pulled at round t, and i^t_m is the true arm pulled at round t. We also assume that if $data^t_{\mathsf{Controller}}$ does not correspond to an actual collection of encrypted messages that Controller sees, then the advantage for

such an input is negligible. We next show that by using the adversary Adv, we can construct an adversary B able to break the IND-CPA property of AES-GCM. To do so, B creates a simulation of an Samba execution, similarly to the proof of Theorem 5. Even though the messages of such a simulation are encrypted, B knows the keys hence the data of each DO_i. In particular, B knows in clear the messages sent at Step 5 by $\mathsf{Controller}$ to each DO_i at a given time step t. To recall, this message contains the pulling bit b_i. We assume here that there is only one computation round i.e., $nb_{\mathcal{A}} = 1$ and that $b_i = 1$ means that the arm i will be pulled. As input for the IND-CPA game, B sends $m_1 = b_i = 1$ and another message $m_0 = b_{j\neq i} = 0$. Then, B receives back $\mathsf{Enc}(m_b)$, where b is a random bit selected uniformly by the challenger. Next, B calls $\mathsf{Adv}^{pa(t)}(data'_{\mathsf{Controller}})$, where $data'_{\mathsf{Controller}}$ is the collection of encrypted messages from the B's simulation, except that it replaces $\mathsf{Enc}(m_1)$ by $\mathsf{Enc}(m_b)$. The strategy of B is: if Adv returns the correct i_m, then B returns 1, otherwise answer randomly.

- If $b = 0$ (probability $\frac{1}{2}$), then Adv does not receive a correct simulation because no arm is pulled at round t. According to our assumption, Adv does not give any advantage.
  - If Adv returns the correct i_m (probability $\frac{1}{K}$), then B answers 1 and is wrong.
  - Otherwise (probability $1 - \frac{1}{K}$), then B answers randomly and is correct with probability $\frac{1}{2}$. This branch yields a probability of success of $\frac{1}{2}(1 - \frac{1}{K})\frac{1}{2}$.
- If $b = 1$ (probability $\frac{1}{2}$), then the advantage given by Adv can be leveraged by B.
  - If Adv returns the correct i^t_m (probability $\frac{1}{K} + x + \mathrm{negl}(\lambda)$), then B correctly answers 1. This branch's success probability is $\frac{1}{2}(\frac{1}{K} + x + \mathrm{negl}(\lambda))$.
  - Otherwise (probability $1 - \frac{1}{K} - x - \mathrm{negl}(\lambda)$), B answers randomly and is correct with probability $\frac{1}{2}$. This branch yields a probability of success of $\frac{1}{2}(1 - \frac{1}{K} - x - \mathrm{negl}(\lambda))\frac{1}{2}$.

By aggregating the aforementioned cases, the probability α of success of B is:

$$\begin{aligned}\alpha &= \frac{1}{2}(1 - \frac{1}{K})\frac{1}{2} + \frac{1}{2}(\frac{1}{K} + x + \mathrm{negl}(\lambda)) + \frac{1}{2}(1 - \frac{1}{K} - x - \mathrm{negl}(\lambda))\frac{1}{2} \\ &= \frac{1}{4} - \frac{1}{4K} + \frac{1}{2K} + \frac{x}{2} + \frac{1}{4} - \frac{1}{4K} - \frac{x}{4} + \mathrm{negl}(\lambda) = \frac{1}{2} + \frac{x}{4} + \mathrm{negl}(\lambda)\end{aligned}$$

Hence, B has an advantage of $\frac{x}{4}$ in the IND-CPA game, which is non negligible. This contradicts the fact that AES-GCM is IND-CPA secure. Hence, we conclude that there does not exist any PPT adversary Adv that violates the property stated in the theorem. □

B.3 Security of an External Observer

The proofs for the external observer are very similar to the ones for $\mathsf{Controller}$ cf. Section B.2.

Theorem 7. *For an arm $i \in [\![K]\!]$ and a time step $t \in [\![N]\!]$, an honest-but-curious external observer cannot guess $s_{i,t}$, given $data^t_{ext}$, with a probability better than random. More precisely, for all PPT adversaries Adv,*

$$\left|\Pr\left[(i, \hat{s}_{i,t}) \leftarrow \mathsf{Adv}^{sum(t)}(data^t_{ext}); \hat{s}_{i,t} = s_{i,t}\right] - p_Q(t, s_{i,t})\right|$$

is negligible in λ, where $\mathsf{Adv}^{sum(t)}(data^t_{ext})$ returns $(i, \hat{s}_{i,t})$ in which $\hat{s}_{i,t}$ is Adv's guess on $s_{i,t}$ for the arm i (chosen by Adv), and $p_Q(t, s_{i,t})$ is the probability of obtaining a sum of rewards $s_{i,t}$ from at most t pulls of arm i until round t.

Proof. The external observer collects $data^t_{ext}$, which consists of several encrypted messages, some of them being encrypted with Enc (AES-GCM) and some other being encrypted with $\mathcal{E}_{\mathsf{DC}}$ (Paillier). We prove that these messages do not provide an advantage bigger than the advantage of an adversary in a classical IND-CPA game on Enc or $\mathcal{E}_{\mathsf{DC}}$. For simplicity, we assume that the $data^t_{ext}$ only contains two encrypted messages, $\mathsf{Enc}(m)$ and $\mathcal{E}_{\mathsf{DC}}(n)$. The proof can obviously be adapted if $data^t_{ext}$ consists of more than two messages.

The goal of the adversary is to extract at least a bit of information from either m or n. The entropy of this system is minimal when $m = n$. Hence, when $m = n$, the adversary has the highest probability of guessing at least a bit from either m or n (which are the same in this case). As a consequence, in the general case, the advantage of an adversary having to guess a bit about m or n, knowing $\mathsf{Enc}(m)$ or $\mathcal{E}_{\mathsf{DC}}(n)$ is bounded above by the advantage of an adversary having to guess a bit about m, knowing $\mathsf{Enc}(m)$ and $\mathcal{E}_{\mathsf{DC}}(m)$.

We prove that the advantage of a PPT adversary in this latter case (having to guess a bit about m from $\mathsf{Enc}(m)$ and $\mathcal{E}_{\mathsf{DC}}(m)$) is negligible. We assume, toward a contradiction, that there exists a PPT adversary Adv able to win the game where, given $\mathsf{Enc}(m)$ and $\mathcal{E}_{\mathsf{DC}}(m)$, Adv recovers a bit of information about m with a non-negligible advantage x: given $\mathsf{Enc}(m)$ and $\mathcal{E}_{\mathsf{DC}}(m)$, the probability that Adv outputs a correct guess about a bit of m is equal to $\frac{1}{2} + x + \mathrm{negl}(\lambda)$. We use this adversary to create another adversary B able to break the IND-CPA property of the encryption schemes Enc (or $\mathcal{E}_{\mathsf{DC}}$, respectively). As usually in the IND-CPA game, B chooses two messages m_0 and m_1, and sends them to the challenger. Then, B receives the challenge $\mathsf{Enc}(m_b)$ (or $\mathcal{E}_{\mathsf{DC}}(m_b)$, respectively), and calls $\mathsf{Adv}(\mathsf{Enc}(m_b), \mathcal{E}_{\mathsf{DC}}(m_0))$ (or $\mathsf{Adv}(\mathcal{E}_{\mathsf{DC}}(m_b), \mathsf{Enc}(m_0))$, respectively). If Adv returns a correct guess about m_0, then B returns 0. Otherwise, it returns 1.

- If $b = 0$ (happens with probability $\frac{1}{2}$), then Adv has a non negligible advantage in guessing a bit about m.
  - Adv outputs a correct guess about one bit of m_0 with probability $\frac{1}{2} + x + \mathrm{negl}(\lambda)$. In this case, B is correct. This branch happens with probability $\frac{1}{2}(\frac{1}{2} + x + \mathrm{negl}(\lambda))$.
  - If Adv does not answer correctly (happens with probability $\frac{1}{2} - x - \mathrm{negl}(\lambda)$), then Adv is correct with probability $\frac{1}{2}$. This branch happens with probability $\frac{1}{2}(\frac{1}{2} - x - \mathrm{negl}(\lambda))\frac{1}{2}$.
- If $b = 1$ (happens with probability $\frac{1}{2}$), then Adv has no advantage.
  - If Adv returns a correct guess about one bit of m_0 (happens with probability $\frac{1}{2}$), then B is wrong.
  - If not (happens with probability $\frac{1}{2}$), then Adv returns a random guess and is correct with probability $\frac{1}{2}$. This branch of events happen with probability $\frac{1}{2^3}$.

By aggregating these cases, the probability α of success of B is:

$$\begin{aligned}\alpha &= \frac{1}{2}(\frac{1}{2} + x + \mathrm{negl}(\lambda)) + \frac{1}{2}(\frac{1}{2} - x - \mathrm{negl}(\lambda))\frac{1}{2} + \frac{1}{8} \\ &= \frac{1}{4} + \frac{x}{2} + \frac{1}{8} - \frac{x}{4} + \frac{1}{8} + \mathrm{negl}(\lambda) = \frac{1}{2} + \frac{x}{4} + \mathrm{negl}(\lambda)\end{aligned}$$

Hence, B has a non-negligible advantage of $\frac{x}{4}$ in the IND-CPA game against Enc (or $\mathcal{E}_{\mathsf{DC}}$, respectively), which is a contradiction with its IND-CPA property. Guessing a bit about the encrypted message is equivalent to guessing the reward with a probability better than random (i.e., better than $p_Q(t, s_{i,t})$ cf. theorem statement), which concludes the proof. □

As a corollary, by Lemma 2 and Theorem 7, we infer that the external observer cannot guess the cumulative reward with probability better than random.

Theorem 8. *At each time step $t \in [\![K+1, N]\!]$, an honest-but-curious external observer cannot guess which arm is pulled at round t, given $data^t_{ext}$, with probability better than random. More precisely, for all PPT adversaries* Adv*,*

$$\left| \Pr[\mathsf{Adv}^{pa(t)}(data^t_{ext}) = i^t_m] - \frac{1}{K} \right| \textit{ is negligible in } \lambda,$$

where $\mathsf{Adv}^{pa(t)}(data^t_{ext})$ *returns the guess of* Adv *on which arm is pulled at round t, and i^t_m is the true arm pulled at round t.*

Proof. The proof of this theorem is similar to the proof of Theorem 6, at the difference that we consider $data^t_{ext}$ all data known by the external observer at the end of a time step $t \in [\![K+1, N]\!]$, which is composed of messages, encrypted either by AES-GCM either with the Paillier's cryptosystem. The reduction remains unchanged. □

Appendix C. Omitted Details about Experiments

We give a few complements to the experimental setting introduced in Section 6.1. Our preprocessing follows existing techniques (Kohli et al., 2013)(Ciucanu et al., 2020)(Kuleshov & Precup, 2014)(Shi & Shen, 2021) and relies on thresholds 4 for MovieLens and 4.5 for Jester. Moreover, some cumulative reward maximization algorithms have parameters cf. Figure 4. For instance, the ε-greedy algorithm needs an $\varepsilon \in [0, 1]$. When ε tends to zero, ε-greedy acts like a pure greedy algorithm. At the opposite, when ε tends to 1, ε-greedy choose randomly an arm uniformly. We tuned the fixed ε by trying different values $\{0, 0.1, 0.2, ..., 1\}$ and choosing the one that gives the best cumulative rewards, that is $\varepsilon = 0.1$ for both MovieLens and Jester. For ε-decreasing-greedy, ε evolves over time according to the function $f_\varepsilon = \frac{1}{\ln(t)}$. Using similar techniques, for Softmax, we run the algorithm with $\tau \in \{0.0, 0.01, 0.02, ..., 0.1\}$. We obtained $\tau = 0.06$ for MovieLens and $\tau = 0.02$ for Jester. Furthermore, for Pursuit, we run the algorithm with $\beta \in \{0, 0.1, 0.2, ..., 1\}$. We obtained $\beta = 0.1$ for MovieLens and $\beta = 0.2$ for Jester.