Encrypted Linear Contextual Bandit

Evrard Garcelon
Meta AI & CREST, ENSAE

Vianney Perchet
CREST, ENSAE

Matteo Pirotta
Meta AI

Abstract

Contextual bandit is a general framework for online learning in sequential decision-making problems that has found application in a wide range of domains, including recommendation systems, online advertising, and clinical trials. A critical aspect of bandit methods is that they require to observe the contexts –i.e., individual or group-level data– and rewards in order to solve the sequential problem. The large deployment in industrial applications has increased interest in methods that preserve the users' privacy. In this paper, we introduce a privacy-preserving bandit framework based on homomorphic encryptionwhich allows computations using encrypted data. The algorithm *only* observes encrypted information (contexts and rewards) and has no ability to decrypt it. Leveraging the properties of homomorphic encryption, we show that despite the complexity of the setting, it is possible to solve linear contextual bandits over encrypted data with a $\widetilde{O}(d\sqrt{T})$ regret bound in any linear contextual bandit problem, while keeping data encrypted.

1 INTRODUCTION

Contextual bandits have become a key part of several applications such as marketing, healthcare and finance; as they can be used to provide personalized e.g., adaptive service (Bastani and Bayati, 2020; Sawant et al., 2018). In such application, algorithms receives as input users' features, i.e. the "contexts", to tailor their recommendations. Those features may disclose sensitive information, as personal (e.g., age, gender, etc.) or geo-localized features are commonly used in recommendation systems. Privacy awareness has increased over years and users are less willing to disclose information and are more and more concerned about how their personal data is used (Das et al., 2021). For example, a user may be willing to receive financial investment suggestion but not to share information related to income, deposits, properties owned and other assets. However, without observing this important information about a user, a service provider may not be able to provide meaningful investment guidance to the user. This example extends to many other applications. For instance, suppose an user is looking for a restaurant nearby, if the provider has no access to even a coarse geo-location, it would not be able to provide meaningful suggestions to the user. An effective approach to address these concerns is to resort to end-to-end encryption to guarantee that data is readable only by the users (Kattadige et al., 2021). In this scenario, the investment company or the service provider observes only an encrypted version of user's information and have no ability to decrypt it. While this guarantee high level of privacy, it is unclear whether the problem remains learnable and how to design effective online learning algorithms in this secure scenarios.

In this paper, we introduce - and analyze - the setting of encrypted contextual bandit to model the mentioned scenarios. At each round, a bandit algorithm observes encrypted features (including e.g., geo-location, food preferences, visited restaurants), chooses an action (e.g., a restaurant) and observes an encrypted reward (e.g., user's click), that is used to improve the quality of recommendations. While it is possible to obtain end-to-end encryption –i.e., the bandit algorithm only observes encrypted information that is not able to decrypt– using standard encryption methods (e.g., AES, RSA, TripleDES), the provider may no longer be able to provide a meaningful service since may not be able to extract meaningful information from encrypted features. We thus address the following question:

Is it possible to learn with encrypted contexts and rewards? And what is the associated computational and learning cost?

Homomorphic Encryption (Halevi, 2017, HE) is a pow-

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

erful encryption method that allows to carry out computation of encrypted numbers. While this is a very powerful idea, only a limited number of operations can be performed, notably only addition and/or multiplication. While HE has been largely investigated in supervised learning (Badawi et al., 2020; Graham, 2015), little is known about online learning. In this paper we aim to look into this direction. We approach the aforementioned question via HE and from a theoretical point-of-view. We consider the case of linear rewards and investigate the design of a "secure" algorithm able to achieve sub-linear regret in this setting. There are several challenges in the design of bandit algorithms that makes the application of HE techniques not easy. First, it is not obvious that all the operations required by a bandit algorithm (notably optimism) can be carried out only through additions and multiplications. Second, errors or approximations introduced by the HE framework to handle encrypted data may compound and prevent to achieve provably good performance. Finally, a careful algorithmic design is necessary to limit the total number of HE operations, which are computationally demanding.

Contributions. Our main contributions can be summarized as follows: **1)** We introduce and formalize the problem of secure contextual bandit with homomorphic encryption. **2)** We provide the first bandit algorithm able to learn over encrypted data in contextual linear bandits, a standard framework that allows us to describe and address all the challenges in leveraging HE in online learning. Leveraging optimism (e.g., Abbasi-Yadkori et al., 2011) and HE, we introduce HELBA which balances security, approximation error due to HE and computational cost to achieve a $\widetilde{O}(\sqrt{T})$ regret bound. This shows that i) it is possible to learn *online* with encrypted information; ii) preserving users' data security has negligible impact on the learning process. This is a large improvement w.r.t. ε-LDP which has milder security guarantees and where the best known bound is $\widetilde{O}(T^{3/4}/\varepsilon)$. **3)** We discuss practical limitations of HE and ways of improving the efficiency of the proposed algorithm, mainly how the implementation of some procedures can speed up computations and allow to scale dimension of contexts. We report preliminary numerical simulations confirming the theoretical results.

Related work. To prevent information leakage, the bandit literature has mainly focused on Differential Privacy (DP) (e.g., Shariff and Sheffet, 2018; Tossou and Dimitrakakis, 2016). While standard (ϵ, δ)-DP enforces statistical diversity of the output of an algorithm, it does not provide guarantees on the security of user data that can be accessed directly by the algorithm. A stronger privacy notion, called local DP, requires data being privatized before being accessed by the algorithm. While it may be conceptually similar to encryption, i) it does not provide the same security guarantee as encryption (having access to a large set of samples may allow some partial denoising Cheu et al. (2021)); and ii) it has a large impact on the regret of the algorithm. For example, Zheng et al. (2020) recently analyzed ε-LDP in contextual linear bandit and derived an algorithm with $\widetilde{O}(T^{3/4}/\varepsilon)$ regret bound to be compared with a $\widetilde{O}(\sqrt{T})$ regret of non-private algorithms. Homomorphic Encryption (e.g. Halevi, 2017) has only been merely used to encrypt rewards in bandit problems (Ciucanu et al., 2020, 2019), but in some inherently simpler setting than the setting considered here (see App. B).

2 HOMOMORPHIC ENCRYPTION

Homomorphic Encryption (Halevi, 2017) is a *probabilistic encryption method* that enables an untrusted party to perform some computations (addition and/or multiplication) on encrypted data. Formally, given two original messages m_1 and $m_2 \in \mathbb{R}$, the addition (resp. multiplication) of their encrypted versions (called ciphertexts) is equal to the encryption of their sum $m_1 + m_2$ (resp. $m_1 \times m_2$), hence the name "homomorphic".[1] We consider a generic homomorphic schemes that generate a public key **pk** (distributed widely and used to encrypt messages), and private keys **sk** (used for decryption of encrypted messages). This private key is, contrary to the public key, obviously assumed to be kept private.

More precisely, we shall consider *Leveled Fully Homomorphic* encryption (LFHE) schemes for real numbers. This type of schemes supports both additions and multiplications but only for a fixed and finite number of operations, referred to as the *depth*. This limitation is a consequence of HE's probabilistic approach. Although noisy encryption allows to achieve high security, after a certain number of operations the data is drown in the noise (e.g., Albrecht et al., 2015), resulting in an indecipherable ciphertext (the encrypted message). In most LFHE schemes, the depth is the maximum number of operations possible before losing the ability to decrypt the message. Often multiplications have a significantly higher noise growth than addition and the depth refers to the maximum number of multiplication between ciphertexts possible. The security of a LFHE schemes is defined by $\kappa \in \mathbb{N}$, usually $\kappa \in \{128, 192, 256\}$. A κ-bit level of security means that an attacker has to perform roughly 2^κ operations to break the encryption scheme,

[1] Most schemes also support Single Instruction Multiple Data (SIMD), i.e., the same operation on multiple data points in parallel.

i.e., to decrypt a ciphertext without the secret key.

Formally, an LFHE scheme is defined by:

- *A key generator function $KeyGen(N, D, \kappa)$*: takes as input the maximum depth D (e.g., max. number of multiplications), a security parameter κ and the degree N of polynomials used as ciphertexts (App. C.1). It outputs a secret key $\mathbf{sk}$ and a public key $\mathbf{pk}$.
- *An encoding function $Enc_{\mathbf{pk}}(m)$*: encrypts the message $m \in \mathbb{R}^d$ with the public key $\mathbf{pk}$. The output is a ciphertext $\mathbf{ct}$, a representation of m in the space of complex polynomials of degree N.
- *A decoding function $Dec_{\mathbf{sk}}(\mathbf{ct})$*: decrypts the ciphertext $\mathbf{ct}$ of $m \in \mathbb{R}^d$ using the secret key $\mathbf{sk}$ and outputs message m.
- *An additive operator $Add(\mathbf{ct}_1, \mathbf{ct}_2)$*: for ciphertexts $\mathbf{ct}_1$ and $\mathbf{ct}_2$ of messages m_1 and m_2, it outputs ciphertext $\mathbf{ct}_{add}$ of $m_1 + m_2$: $Dec_{\mathbf{sk}}\Big(Add\big(Enc_{\mathbf{pk}}(m_1), Enc_{\mathbf{pk}}(m_2)\big)\Big) = m_1 + m_2$.
- *A multiplicative operator $Mult(\mathbf{ct}_1, \mathbf{ct}_2)$*: similar to *Add* but for ciphertexts $\mathbf{ct}_1$ and $\mathbf{ct}_2$ of messages m_1 and m_2 and output ciphertext $\mathbf{ct}_{mult}$ of $m_1 \cdot m_2$.

To avoid to complicate the notation we will use classical symbols to denote addition and multiplication between ciphertexts. Choosing D as small as possible is essential, as it is the major bottleneck for performance, in particular at the keys generation step. This cost comes from the fact that the dimension of a ciphertext N needs to grow with D for a given security level κ: namely $N \geq \Omega(\kappa D)$ (refer to App. C.1 for more details). In this paper, we choose to use the CKKS scheme (Cheon et al., 2017) because it supports operations on real numbers.

Other HE schemes. Most HE schemes (ElGamal, 1985; Paillier, 1999; Rivest et al., 1978) are *Partially Homomorphic* and only support either additions or multiplications, but not both. Other schemes that support any number of operations are called *Fully Homomorphic* encryption (FHE) schemes. Most LFHE schemes can be turned into FHE schemes thanks to the bootstrapping technique introduced by Gentry and Boneh (2009). However, the computational cost is extremely high. It is thus important to optimize the design of the algorithm to minimize its multiplicative depth and (possibly) avoid bootstrapping (Acar et al., 2018; Ducas and Micciancio, 2015; Zhao and Wang, 2018).

3 CONTEXTUAL BANDIT AND ENCRYPTION

A contextual bandit problem is a sequential decision-making problem with $K \in \mathbb{N}_+$ arms and horizon $T \in \mathbb{N}_+$ (e.g., Lattimore and Szepesvári, 2020). At each time $t \in [T] := \{1, \dots, T\}$, a learner first observes a set of features $(s_{t,a})_{a\in[K]} \subset \mathbb{R}^d$, selects an action $a_t \in [K]$ and finally observes a reward $r_t = r(s_{t,a_t}) + \eta_t$ where η_t is a conditionally independent zero-mean noise. We do not assume anything on the distribution of the features $(s_{t,a})_a$. The performance of the learner $\mathfrak{A}$ over T steps is measured by the regret, that measures the cumulative difference between playing the optimal action and the action selected by the algorithm. Formally, let $a_t^\star = \arg\max_{a\in[K]} r(s_{t,a})$ be the optimal action at step t, then the pseudo-regret is defined as:

$$R_T = \sum_{t=1}^{T} r(s_{t,a_t^\star}) - r(s_{t,a_t}). \qquad (1)$$

To protect privacy and avoid data tempering, we introduce end-to-end encryption to this protocol. Contexts and rewards are encrypted before being observed by the learner; we call this setting encrypted contextual bandit (Alg. 1). Formally, at time $t \in [T]$, the learner $\mathfrak{A}$ observes encrypted features $x_{t,a} = Enc_{\mathbf{pk}}(s_{t,a})$ for all actions $a \in \mathcal{A}$, and the encrypted reward $y_t = Enc_{\mathbf{pk}}(r_t)$ associated to the selected action a_t. The learner may know the public key $\mathbf{pk}$ but not the secure key $\mathbf{sk}$. The learner is thus not able to decrypt messages and it *never* observes the true contexts and rewards. We further assume that both the agent $\mathfrak{A}$ and the users follow the honest-but-curious model, that is to say each parties follow their protocol honestly but try to learn as much as possible about the other parties private

Algorithm 1 Encrypted Contextual Bandit (Server-Side)

```
Input: Agent: $\mathfrak{A}$, public key: $\mathbf{pk}$, horizon: $T$
for $t = 1, \dots, T$ do
    Agent $\mathfrak{A}$ observes encrypted context $(x_{t,a})_{a\in[K]} = (Enc_{\mathbf{pk}}(s_{t,a}))_{a\in[K]}$
    Agent $\mathfrak{A}$ computes the next action as a function of the encrypted history and $(x_{t,a})_{a\in[K]}$ and outputs an encrypted action $u_t = Enc_{\mathbf{pk}}(a_t)$
    Agent $\mathfrak{A}$ observes encrypted reward $y_t = Enc_{\mathbf{pk}}(r_t)$
end for
```

Algorithm 2 Encrypted Contextual Bandit (User-Side)

```
Input: public key: $\mathbf{pk}$, secret key: $\mathbf{sk}$
for $t = 1, \dots, T$ do
    User $t$ observes features $(s_{t,a})_{a\leq K}$ and sends $(x_{t,a})_{a\in[K]} = (Enc_{\mathbf{pk}}(s_{t,a}))_{a\in[K]}$ to the server
    User $t$ receives encrypted action $u_t$
    User $t$ decrypts action $a_t = Dec_{\mathbf{sk}}(u_t)$
    User $t$ observes reward $r_t = r(s_{t,a_t}) + \eta_t$ and sends $Enc_{\mathbf{pk}}(r_t)$ to the server
end for
```

data. [2] As a consequence, the learner can only do computation on the encrypted information. As a result, all the internal statistics used by the bandit algorithm are now encrypted. On user's side (see Alg. 2), upon receiving an encrypted action $u_t = Enc_{\mathbf{pk}}(a_t)$ and decrypting it $a_t = Dec_{\mathbf{sk}}(u_t)$ using the secure key **sk**, the user generates a reward $r_t = r(s_{t,a_t}) + \eta_t$ and sends to the learner the associated ciphertext y_t. The learning algorithm is able to encrypt the action since the public key is publicly available. See App. C for additional details.

We focus on the well-known linear setting where rewards are linearly representable in the features. Formally, for any feature vector $s_{t,a}$, the reward is $r(s_{t,a}) = \langle s_{t,a}, \theta^\star \rangle$, where $\theta^\star \in \mathbb{R}^d$ is unknown. For the analysis, we rely on the following standard assumption:

Assumption 1. *There exists $S > 0$ such that $\|\theta^\star\|_2 \leq S$ and there exists $L \geq 1$ such that, for all time $t \in [T]$ and arm $a \in [K]$, $\|s_{t,a}\|_2 \leq L$ and $r_t = \langle s_{t,a}, \theta^\star \rangle + \eta_t \in [-1, 1]$ with η_t being σ-subGaussian for some $\sigma > 0$.*

4 AN ALGORITHM FOR ENCRYPTED LINEAR CONTEXTUAL BANDITS

In the previous section, we have introduced a generic framework for contextual bandit with encrypted information. Here, we provide the first algorithm able to learn with encrypted observations. In the non-secure protocol, algorithms based on the optimism-in-the-face-of-uncertainty (OFU) principle such as LinUCB (Chu et al., 2011) and OFUL (Abbasi-Yadkori et al., 2011) have been proved to achieve the regret bound $O\big(Sd\sqrt{T}\ln(TL)\big)$. Clearly, they will fail to be used as is in the secure protocol and need to be re-thinked around the limitations of HE (mainly approximations in most operations). As mentioned in the introduction, there are many, both theoretical and practical, challenges to leverage HE in this setting. Indeed, **1)** computing an estimate of the parameter $\theta^\star$ from ridge regression is extremely difficult with HE as finding the inverse of a matrix is not directly feasible for a leveled scheme (Esperança et al., 2017). **2)** Similarly, computing the bonus for the optimistic action selection requires invoking operations that are not naturally available in HE hence incurring a large computational cost. Finally, **3)** computing the maximum element (or maximum index) of a list of encrypted values is non-trivial for the algorithm alone, as it cannot observe the values to compare. In this section, we will provide HE compatible operations addressing these three issues. Each step is highly non-trivial and correctly combining them is even more challenging due to error compounding. We believe the solution we provide for each individual step may be of independent interest.

Algorithm 3 Simplified HELBA

```
for $t = 1, \dots, T$ do
    if Update (Step ❹) then
        Step ❶: Estimate encrypted parameter using $\{x_{l,a_l}, y_l\}_{l \in [t-1]}$
    end if
    Observe encrypted contexts $(x_{t,a})_{a \in [K]} = (Enc_{\mathbf{pk}}(s_{t,a}))_{a \in [K]}$
    Step ❷: Compute encrypted indexes $(\rho_a(t))_{a \in [K]}$
    Step ❸: Compute $\arg\max_a \{\rho_a(t)\}$
end for
```

Alg. 3 report a simplified version of our HE bandit algorithm. Informally, at each round t, our algorithm HELBA (*Homomorphically Encrypted Linear Bandits*) builds an HE estimate ω_t of the unknown $\theta^\star$ ($\omega^\star = Enc_{\mathbf{pk}}(\theta^\star)$) using the observed encrypted samples, compute HE optimistic indexes $(\rho_a(t))_a$ for each action and select the action maximizing the index. We stress that all the mentioned statistics (ω_t and $\rho_a(t)$) are encrypted values. Indeed, HELBA operates directly in the encrypted space, i.e., the space of *complex polynomials* of degree N. Let's analyze those three steps.

Step ❶: HE Friendly Ridge Regression

The first step is to build an estimate of the parameter $\theta^\star$. In the non-encrypted case, we can simply use $\theta_t = V_t^{-1} \sum_{l=1}^{t-1} s_{l,a_l} r_l$, where $V_t = \sum_{l=1}^{t-1} s_{l,a_l} s_{l,a_l}^T + \lambda I$. With encrypted values $(x_{l,a_l}, y_l)_{l \in [t-1]}$, it is possible to compute an encrypted matrix $\Lambda_t = \sum_{l=1}^{t-1} x_{l,a_l} x_{l,a_l}^T + \lambda Enc_{\text{pk}}(I) = Enc_{\text{pk}}(V_t)$ and vector $\sum_{l=1}^{t-1} x_{l,a_l} y_l$ as these operations (summing and multiplying) are HE compatible. The issue resides in the computation of Λ_t^{-1}. An approximate inversion scheme can be leveraged though.

Given a matrix $V \in \mathbb{R}^d$ with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d > 0$ and $c \in \mathbb{R}$ such that for all $i \in [d]$, $\lambda_i \in \text{Conv}\left(\{z \in \mathbb{R} \mid |z - c| \ \leq c\}, 2c\right) \backslash \{0, 2c\}$[3], we define the following sequence of matrices (Guo and Higham, 2006)

$$X_{k+1} = X_k(2I_d - M_k), \quad M_{k+1} = (2I_d - M_k)M_k, \quad (2)$$

initialized at $X_0 = \frac{1}{c} I_d$ and $M_0 = \frac{1}{c} V$. We can show that this sequence converges to V^{-1}.

Proposition 2. *If $V \in \mathbb{R}^{d \times d}$ is a symmetric positive definite matrix, $c \geq Tr(V)$ and for some precision level*

[2] A trusted third party can be used to generate a public and secret keys. Those keys are then sent to the users but **not** to the agent $\mathfrak{A}$ (see Sec. 6).

[3] Conv(E) is the convex hull of set E.

$\varepsilon > 0$, the iterate in (2) satisfies $\|X_k - V^{-1}\| \le \varepsilon$ for any $k \ge k_1(\varepsilon)$ with $k_1(\varepsilon) = \frac{1}{\ln(2)} \ln\left(\frac{\ln(\lambda)+\ln(\varepsilon)}{\ln\left(1-\frac{\lambda}{c}\right)}\right)$, where $\lambda \le \lambda_d$ is a lower bound to the minimal eigenvalue of V and $\|\cdot\|$ is the matrix spectral-norm.

Since V_t is a regularized matrix, it holds that $\lambda_d \ge \lambda > 0$ and by setting $c = \lambda d + L^2 t$ we get that $c \ge \mathrm{Tr}(V_t) \ge \max_i\{\lambda_i\}$, for any step $t \in [T]$. Therefore, we can apply iterations (2) to $\Lambda_t = Enc_{\mathrm{pk}}(V_t)$ since are all HE compatible operations (additions and matrix multiplications). For $\varepsilon_t > 0$, iterations (2) gives a ε_t-approximation $A_t := X_{k_1(\varepsilon_t)}$ of V_t^{-1}, i.e., $\|Dec_{\mathrm{sk}}(A_t) - V_t^{-1}\| \le \varepsilon_t$. As a consequence, an encrypted estimate of the unknown parameter $\theta^\star$ can be computed by mere simple matrix multiplications $\omega_t = A_t \sum_{l=1}^{t-1} x_{l,a_l} y_l$. Leveraging the concentration of the inverse matrix, the following error bound for the estimated parameter holds.

Corollary 3. *Setting $\varepsilon_t = \left(L t^{3/2}\sqrt{L^2 t + \lambda}\right)^{-1}$ in Prop. 2, then $\|Dec_{\mathbf{sk}}(\omega_t) - \theta_t\|_{V_t} \le t^{-1/2}$, $\forall t$.*

This result, along with the standard concentration for linear bandit (Abbasi-Yadkori et al., 2011, Thm. 2), implies that, at all time steps t, with probability at least $1-\delta$:

$$\theta^\star \in \widetilde{\mathcal{C}}_t := \{\ \theta \in \mathbb{R}^d \mid \|Dec_{\mathbf{sk}}(\omega_t) - \theta\|_{V_t} \le \widetilde{\beta}_t\}, \quad (3)$$

where $\|a\|_B = \sqrt{a^\top B a}$ and $\widetilde{\beta}_t = t^{-1/2} + S\sqrt{\lambda} + \sigma\sqrt{d\left(\ln\left(1 + L^2 t/\lambda\right) + \ln(\pi^2 t^2/(6\delta))\right)}$ is the inflated confidence interval due to the approximate inverse (see Prop. 9 in App. D.4). Note that $\widetilde{\beta}_t$ is a plain scalar, not an encrypted value.

Step ❷: Computing The Optimistic Index

Once solved the encrypted ridge regression, the next step for HELBA is to compute an optimistic index $\rho_a(t)$ such that $r(s_{t,a}) \lessapprox Dec_{\mathbf{sk}}(\rho_a(t))$. For any feature vector $s_{t,a}$, by leveraging the confidence interval in (3), the optimistic (unencrypted) index is given by $\max_{\theta \in \widetilde{\mathcal{C}}_t} \langle \theta, s_{t,a}\rangle = \langle Dec_{\mathbf{sk}}(\omega_t), s_{t,a}\rangle + \widetilde{\beta}_t \|s_{t,a}\|_{V_t^{-1}}$. Leveraging Prop. 2, the definition of ε_t in Cor. 3 and $\|s_{t,a}\|_2 \le L$, it holds that:

$$\forall s_{t,a},\ \|s_{t,a}\|^2_{V_t^{-1}} - \|s_{t,a}\|^2_{Dec_{\mathbf{sk}}(A_t)} \le L^2 \|V_t^{-1} - Dec_{\mathbf{sk}}(A_t)\| \le L t^{-\frac{3}{2}} (\lambda + L^2 t)^{-1/2}$$

which leads to $\max_{\theta \in \widetilde{\mathcal{C}}_t} \langle \theta, s_{t,a}\rangle \le \langle Dec_{\mathbf{sk}}(\omega_t), s_{t,a}\rangle + \sqrt{\|s_{t,a}\|^2_{Dec_{\mathbf{sk}}(A_t)} + L\left(t^{3/2}\sqrt{\lambda + L^2 t}\right)^{-1}}$. As a consequence, we can write that the encrypted optimistic index is given by:

$$\rho_a(t) \approx \langle \omega_t, x_{t,a}\rangle + \widetilde{\beta}_t \times \\ \times \mathrm{sqrt}_{\mathrm{HE}}\Big(\underbrace{x_{t,a}^\top A_t x_{t,a} + L\left(t^{3/2}\sqrt{\lambda + L^2 t}\right)^{-1}}_{\maltese}\Big) \quad (4)$$

where $\mathrm{sqrt}_{\mathrm{HE}}$ is an approximate root operator in the encryption space. Unfortunately, computing the root is a non-native operation in HE and we need to build an approximation of it.

For a real value $z \in [0,1]$, we define the following sequences (Cheon et al., 2020)

$$q_{k+1} = q_k\left(1 - \frac{v_k}{2}\right), \quad v_{k+1} = v_k^2\left(\frac{v_k - 3}{4}\right) \quad (5)$$

where $q_0 = z$ and $v_0 = z - 1$. It is possible to show that this sequence converges to $\sqrt{z}$.

Proposition 4. *For any $z \in \mathbb{R}_+$, $c_1, c_2 > 0$ with $c_2 \ge z \ge c_1$ and a precision $\varepsilon > 0$, let q_k be the result of k iterations of Eq. (5), with $q_0 = \frac{z}{c_2}$ and $v_0 = \frac{z}{c_2} - 1$. Then, $|q_k\sqrt{c_2} - \sqrt{z}| \le \varepsilon$ for any $k \ge k_0(\varepsilon) := \frac{1}{\ln(2)}\left(\ln\left(\ln(\varepsilon) - \ln\left(\sqrt{c_2}\right)\right) - \ln\left(4\ln\left(1 - \frac{c_1}{4c_2}\right)\right)\right)$.*

Therefore, by setting $z = \|x_{t,a}\|^2_{A_t} + c_1$ (i.e., as $\maltese$ in Eq. (4)), $c_1 = L(t^{3/2}\sqrt{\lambda + L^2 t})^{-1}$, $c_2 = c_1 + L^2\lambda^{-1/2}\left(1 + \lambda^{-1/2}\right)$ and $\varepsilon = t^{-1}$, we set

$$\rho_a(t) = \langle \omega_t, x_{t,a}\rangle + \widetilde{\beta}_t\left(\sqrt{c_2}\, q_{k_0(1/t)} + \frac{1}{t}\right), \quad (6)$$

which implies that $r(s_{t,a}) \lessapprox \max_{\theta \in \widetilde{\mathcal{C}}_t} \langle \theta, s_{t,a}\rangle \le Dec_{\mathbf{sk}}(\rho_a(t))$. Note that while ω_t, $x_{t,a}$ and q_i are encrypted values, $\widetilde{\beta}_t$, c_1, c_2 and t are plain scalars.

Step ❸: HE Approximate Argmax

The last challenge faced by the learning algorithm is to compute $\arg\max_{a \in [K]}\{\rho_a(t)\}$. Although, it is theoretically possible to compute an argmax procedure operating on encrypted numbers (Gentry and Boneh, 2009), it is highly non practical because it relies on bootstrapping. Recently, Cheon et al. (2020) introduced an homomorphic compatible algorithm (i.e., approximate), called **NewComp**, that builds a polynomial approximation of $\mathrm{Comp}(a,b) = \mathbb{1}_{\{a>b\}}$ for any $a, b \in [0,1]$. This algorithm allows to compute an HE friendly approximation of $\max\{a,b\}$ for any $a, b \in [0,1]$. We leverage this idea to derive **acomp**, a homomorphic compatible algorithm to compute an approximation of the maximum index (see Alg. 9 in App. D.5). Precisely, **acomp** does not compute $\arg\max_{a \in [K]}\{\rho_a(t)\}$ but an approximate vector $b_t \approxeq (\mathbb{1}_{\{a = \arg\max_i \rho_i(t)\}})_{a \in [K]}$. The maximum index is the value a such that $(b_t)_a$ is greater than a threshold accounting for the approximation error.

The **acomp** algorithm works in two phases. First, **acomp** computes an approximation M of $\max_{i \in [K]}\{\rho_i(t)\}$ by comparing each pair $(\rho_i(t), \rho_j(t))$ with $i < j \le K$. Second, each value $\rho_a(t)$ is compared to this approximated maximum value M to obtain $(b_t)_a$, an approximate computation of $\mathbb{1}_{\{\rho_a(t) > M\}}$.

Cor. 5 shows that if a component of b_t is big enough, the difference between $\max_a \ \rho_a(t)$ and any arm with $4(b_t)_a \geq t^{-1}$ is bounded by $\widetilde{O}(1/t)$ (proof in App. D.5).

Corollary 5. *At any time $t \in [T]$, any arm $a \in [K]$ satisfying $(b_t)_a \geq \frac{1}{4t}$ is such that:*

$$\begin{aligned} \rho_a(t) \geq \max_{a' \in [K]} \{\rho_{a'}(t)\} - \frac{1}{t} \\ - \frac{\widetilde{\beta}_t}{t}\left[\frac{2}{t} + \sqrt{\frac{L}{t^{3/2}\sqrt{\lambda + L^2 t}}} + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}}\right] \end{aligned} \tag{7}$$

Cor. 5 shows that while an action a such that $4t(b_t)_a \geq 1$ may not belong to $\arg\max_{a \in [K]}\{\rho_t(a)\}$, it can be arbitrarily close, hence limiting the impact on the regret. As shown later, this has little impact on the final regret of the algorithm as the approximation error decreases fast enough. Since b_t is encrypted, the algorithm does not know the action to play. b_t is sent to the user who decrypts it and selects the action to play (the user is the only one having access to **sk**). $b_t \approx (\mathbb{1}_{\{a = \max_{i \in [K]} \rho_i(t)\}}$ indicates to the user which action to take which is necessary by design of the bandit problem. However, if the user is able to invert the polynomial functions used to compute b_t thanks to the rescaling of the estimates $(\rho_a(t))_a$ the latter can only learn a relative ranking for this particular user and not the actual estimates.

Step ❹: Update Schedule

Thanks to these steps, we can prove (see App. **??**) a $\sqrt{T}$ regret bound for HELBA when ω_t is recomputed at each step t. However, this approach would be impractical due to the extremely high number of multiplications performed. In fact, inverting the design matrix at each step incurs a large multiplicative depth and computational cost. The most natural way of reducing this cost is to reduce the number of times the ridge regression is solved. The arm selection policy will not be updated at each time step but rather only when necessary. Reducing the number of policy changes is exactly the aim of low switching algorithms (see e.g., Abbasi-Yadkori et al., 2011; Perchet et al., 2016; Bai et al., 2019; Calandriello et al., 2020; Dong et al., 2020). We focus on a dynamic, data-dependent batching since $\sqrt{T}$ regret is not attainable using a fixed known-ahead-of-time schedule (Han et al., 2020).

Abbasi-Yadkori et al. (2011) introduced a low switching variant of OFUL (RSOFUL) that recomputes the ridge regression only when the following condition: $\det(V_{t+1}) \geq (1 + C)\det(V)$ is met, with V the design matrix after the last update. The regret of RSOFUL scales as $\widetilde{\mathcal{O}}(d\sqrt{(1+C)T})$. In the secure setting, computing the determinant of an encrypted matrix is costly (see e.g. Kaltofen and Villard, 2005) and requires multiple matrix multiplications. The complexity of checking the above condition with HE outweighs the benefits introduced by the low switching regime, rendering this technique non practical. Instead of a determinant-based condition, we consider a trace-based condition, inspired by the update rule for GP-BUCB (Desautels et al., 2014; Calandriello et al., 2020).

The "batch j" is defined as the set of time steps between j-th and $(j+1)$-th updates of ω, and we denote by t_j the first time step of this batch. The design matrix is now denoted by $\overline{\Lambda}_j = \lambda Enc_{\mathbf{pk}}(I) + \sum_{l=1}^{t_j-1} x_{l,a_l} x_{l,a_l}^{\intercal}$, and more importantly is only updated at the beginning of each batch j (and similarly for the inverse $\overline{A}_j$ and vector ω_j). The current batch j is ended if and only if the following *trace-based condition* is met at some time t:

$$C \leq \mathrm{Tr}\left(\sum_{l=t_j+1}^{t-1} \overline{A}_j x_{l,a_l} x_{l,a_l}^{\intercal}\right) = \sum_{l=t_j+1}^{t-1} \|x_{l,a_l}\|^2_{\overline{A}_j} \tag{8}$$

The intuition behind this condition is that the trace of $\overline{V}_j = \lambda I + \sum_{l=1}^{t_j-1} s_{l,a_l} s_{l,a_l}^{\intercal}$ is enough to directly control the regret. The following proposition shows that the error due to the computation in the encrypted space remains small.

Proposition 6. *Let $\varepsilon_j = \left(Lt_j^{3/2}\sqrt{\lambda + L^2 t_j}\right)^{-1}$ and $\overline{A}_j = X_{k_1(\varepsilon_j)}$ as in Eq. (2) starting from $M_0 = \overline{\Lambda}_j / c$ with $c \geq \lambda + t_j L^2$. Then, for any $j > 0$: $\left|Tr\Big(\sum_{l=t_j+1}^{t-1}\Big(Dec_{\mathbf{sk}}(A_j) - \overline{V}_j^{-1}\Big) s_{l,a_l} s_{l,a_l}^{\intercal}\Big)\right| \leq L^2 \varepsilon_j (t - 1 - t_j)$.*

Since the switching condition involves data-dependent encrypted quantities, we leverage a similar procedure as to compare indexes. We compute an (*encrypted*) homomorphic approximation of the sign function thanks to the **acomp** algorithm. The result is an encryption of the approximation of $\mathbb{1}_{\{\}}$. Similarly to computing the argmax of $(\rho_a(t))_a$, the algorithm cannot access the result, thus it relies on the user to decrypt and send the result of the comparison to decide whenever the algorithm needs to update the approximate inverse $\bar{A}_j$, . However, to prevent any information leakage, that is to say the algorithm or the user learning about the features of other users, we use a masking procedure which obsfucates the result of the decryption to the user (detailed in App. E.1.1 and App. E.1.2).

In non-encrypted setting, Cond. 8 can be used to dynamically control the growth of the regret, that is bounded by $\mathcal{O}\big(\sum_{j=0}^{M_T}\sum_{t=t_j+1}^{t_{j+1}} \|\overline{V}_j^{-1/2} s_{t,a_t}\|_2\big)$. But in the secure setting, the regret can not be solely bounded as before. The condition for updating the batch has to take into account the approximation error introduced by all the approximate operations. Let M_T be the total number of batches, then the contribution of the approximations to the regret scales as $\sum_{j=0}^{M_T-1} \widetilde{\mathcal{O}}((t_{j+1} - t_j)^2 \varepsilon_j)$.

We thus introduce an additional condition aiming at explicitly controlling the length of each batch. Let $\eta > 0$, then a new batch is started if Cond. (8) is met or if: $t \geq (1+\eta)t_j$. This ensures that the additional regret term grows proportionally to the total number of batches M_T. Note that t_j and t are not encrypted values and the comparison is "simple". The full algorithm is reported in App. A.

5 THEORETICAL GUARANTEES

The regret analysis of HELBA is decomposed in two parts. First, we show that, the number of batches is logarithmic in T. Then, we bound the error of approximations per batch.

Proposition 7. *For any $T > 1$, if $C - \frac{L\eta}{\sqrt{\lambda + L^2}} > \frac{1}{4}$, the number of episodes M_T of* HELBA *(see Alg. 3) is bounded by:*

$$M_T \leq 1 + \frac{d \ln\left(1 + \frac{L^2 T}{\lambda d}\right)}{2 \ln\left(\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}}\right)} + \frac{\ln(T)}{\ln(1+\eta)} \qquad (9)$$

The total number of multiplications to compute ω_j is T/M_T-times smaller thanks to the low-switching condition. This leads to a vast improvement in computational complexity. Note that at each round t, HELBA still computes the upper-confidence bound on the reward and the maximum action. Leveraging this result, when any of the batch conditions is satisfied, the regret can be controlled in the same way as the non-batched case, up to a multiplicative constant.

Theorem 8. *Under Asm. 1, for any $\delta > 0$ and $T \geq d$, there exists constants $C_1, C_2 > 0$ such that the regret of* HELBA *(Alg. 3) is bounded with probability $1 - \delta$ by:*

$$R_T \leq C_1 \beta^\star \left(\sqrt{(1.25 + C)\, dT \ln\left(\frac{TL}{\lambda d}\right)} + \frac{L^{3/2}}{\sqrt{\lambda}} \ln(T) \right)$$
$$+ C_2 \beta^\star M_T \max\left\{ \sqrt{L} + \frac{\eta}{\sqrt{L}}, \eta^2 + \frac{L}{\sqrt{\lambda + L^2}^3} \right\}$$

with $\beta^\star = 1 + \sqrt{\lambda} S + \sigma \sqrt{d\left(\ln\left(1 + \frac{L^2 T}{\lambda d}\right) + \ln\left(\frac{\pi^2 T^2}{6\delta}\right)\right)}$ and M_T as in Prop. 7.

The first term of the regret highlights the impact of the approximation of the square root and maximum that are computed at each round. The second term shows the impact of the approximation of the inverse. It depends on the number of batches since the inverse is updated only once per batch. By Prop. 7, we notice that this term has a logarithmic impact on the regret. Finally, the last term is the regret incurred due to low-switch of the optimistic algorithm. We can notice that the parameter C regulates a trade-off between regret and computational complexity. This term is also the regret incurred by running OFUL with trace condition instead of the determinant-based condition. This further stress that the cost of encryption on the regret is only logarithmic, leading to a regret bound of the same order of the non-secure algorithms. But the computationnal complexity of HELBA is multiple orders higher than any non-encrypted bandit algorithm. For example the complexity of computing a scalar product with HE now scales with the ring dimension N and not the dimension of the contexts anymore $d \ll N$.

6 DISCUSSION AND EXTENSIONS

In this section, we present a numerical validation of the proposed algorithm in a secure linear bandit problem and we discuss limitations and possible extensions.

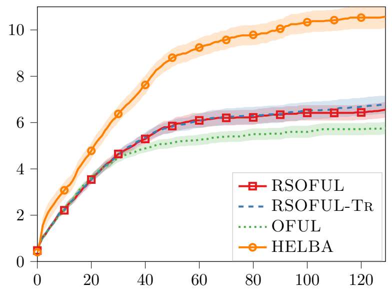

Figure 1: Regret on a toy problem with 4 random uniform contexts.

Numerical simulation. Despite the mainly theoretical focus of the paper, we illustrate the performance of the proposed algorithm on a toy example, where we aim at empirically validating the theoretical findings. We consider a linear contextual bandit problem with 4 contexts in dimension 2 and 2 arms. As baselines, we consider OFUL, RSOFUL and RSOFUL-Tr (a version of RSOFUL where the determinant-based condition is replaced by the trace-condition in (8)). We run these baselines on non-encrypted data and compare the performance with HELBA working with encrypted data. In the latter case, at each step, contexts and rewards are encrypted using the CKKS (Cheon et al., 2017) scheme with parameter $\kappa = 128$, $D = 100$ and $N = 2^{16}$, a modulus $\log(q_0) = 4982$ and a cyclotomic degree of $M = 131072$ chosen automatically by the PALISADE library (PAL, 2020) used for the implementation. The size of the ciphertext is not allowed to grow and a relinearization is performed after every operation. The variance of the noise in the reward is $\sigma = 0.5$. Finally, we use $C = 1$ and $\eta = 0.1$ in HELBA. The regularization parameter is set to 1 and $L = 5.5$. Fig. 1 shows the regret of the algorithms averaged over 25 repetitions. We notice that while the non-encrypted low-switching algorithms (i.e., RSOFUL and RSOFUL-Tr) recompute the ridge regression only 11 times on average, their performance is only slightly affected by this and it is comparable to the one of OFUL. The reduced number of updates is a significant improvement in light of the current limitation in the multiplicative depth of homomorphic schemes. This was the enabling factor to implement HELBA. Note that the update condition in HELBA increases the number of updates to about 20 on average. As expected, the successive approximations and low-switching combined worsen the regret of HELBA. However, this small loss in performance comes with a provable guarantee on the security of users' data.

Computational Complexity. Even though we reduced the number of multiplications and additions, the total runtime of HELBA is still significant, several orders of magnitude higher compared to the unencrypted setting, the total time for $T = 130$ steps and $\kappa = 128$ bits was 20 hours and 39 minutes. We believe that a speed up can be obtained by optimizing how matrix multiplication is handled. For example, implementation optimization can increase the speed of computation of logistic regression (Blatt et al., 2020). However, we stress that HELBA is almost (up to the masking procedure) agnostic to the homomorphic scheme used, hence any improvement in the HE literature can be leveraged by our algorithm. Bootstrapping procedures (Gentry and Boneh, 2009) can be used for converting a leveled schema into a *Fully HE* scheme. This mechanism, together with the low-switching nature of our algorithm, can be the enabling tool for scaling this approach to large problems as the multiplicative depth scales linearly with the dimension.

Discussion. Many other approaches are possible to increase the computational efficiency, for example using a trusted execution environment (Sabt et al., 2015) or leveraging user-side computational capacities. We decided to design an algorithm where the major computation (except for comparisons) are done server-side, having in mind cloud-computing or recommendations running on mobile phone. The objective was to make as secure as possible this protocol so that the server can leverage the information coming from all users. However, if we assume that users have greater computation capabilities, the algorithm can delegate some computations (see e.g., Blatt et al., 2020). For example, for the inverse, the algorithm can generate a random (invertible) matrix N_t, homomorphically compute $V_t N_t$ and sends the masked matrix, $V_t N_t$ to the user. The latter decrypts, inverts, re-encrypts the inverse and sends it to the algorithm (see (Bost et al., 2015, Sec. 8) for more details). A similar scenario, can be imagined for computing a square root or a matrix multiplication. This protocol requires users to perform computationally heavy operations (inverting a matrix) locally. To ensure security with this delegation, a verification step is needed (see e.g., Bost et al., 2015) further increasing communications between the user and the bandit algorithm. We believe that an interesting direction for future work is to integrate this protocol in a distributed setting (i.e., federated learning). Using a server-side trusted execution environment can speed up computations as operations are executed in the clear in private regions of the memory.

Multi-users Setting. Usually contexts represent different users, described by their features s_t and some users may want to use their own public key $\mathbf{pk}_t$ (and secret key $\mathbf{sk}_t$) to encrypt those features. In that case, HELBA can be used with a KeySwitching Fan and Vercauteren; Brakerski (2012); Brakerski et al. (2014) component. This operation takes a ciphertext c_1 decipherable by a secret key $\mathbf{sk}_1$ and output a ciphertext c_2 decipherable by a secret key $\mathbf{sk}_2$. A user send the encrypted context/reward to the bandit algorithm which perform a key switching (see App. C.2) with the help of trusted third party who generate the set of keys used by the learning algorithm such that all ciphertexts received are decipherable by the same key and compatible for homomorphic operations. KeySwitching can be performed without accessing the data and with some (or all) users using their own set of private/public keys for encryption/decryption.

7 CONCLUSION

In this paper, we introduced the problem of encrypted linear contextual bandits and provided an algorithm, HELBA, with a regret similar to regret bounds achievable in the non-encrypted setting. This algorithm trades-off the approximation error and computational complexity of HE and the need for accurate estimation to obtain sublinear regret. We leave as open question the design of an algorithm tailored to the characteristics of the HE and extensions to either other algorithms (e.g., Thompson sampling) or settings (e.g., reinforcement learning).

Acknowledgments

V. Perchet acknowledges support from the French National Research Agency (ANR) under grant number #ANR-19-CE23-0026 as well as the support grant, as well as from the grant "Investissements d'Avenir" (LabEx Ecodec/ANR-11-LABX-0047).

References

PALISADE Lattice Cryptography Library (release 1.10.4). https://palisade-crypto.org/, September 2020.

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24:2312–2320, 2011.

Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), July 2018. ISSN 0360-0300. doi: 10.1145/3214303. URL https://doi.org/10.1145/3214303.

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

Ahmad Al Badawi, Jin Chao, Jie Lin, Chan Fook Mun, Jun Jie Sim, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. Towards the alexnet moment for homomorphic encryption: Hcnn, thefirst homomorphic cnn on encrypted data with gpus, 2020.

Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. Provably efficient q-learning with low switching cost. In *Advances in Neural Information Processing Systems*, pages 8004–8013, 2019.

Hamsa Bastani and Mohsen Bayati. Online decision making with high-dimensional covariates. *Operations Research*, 68(1):276–294, 2020.

Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. *Proceedings of the National Academy of Sciences*, 117 (21):11608–11613, 2020.

Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. 2015.

Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Near-linear time gaussian process optimization with adaptive batching and resparsification. In *International Conference on Machine Learning*, pages 1295–1305. PMLR, 2020.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. Numerical method for comparison on homomorphically encrypted numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 415–445. Springer, 2019.

Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 221–256. Springer, 2020.

Albert Cheu, Adam D. Smith, and Jonathan R. Ullman. Manipulation attacks in local differential privacy. *J. Priv. Confidentiality*, 11(1), 2021.

Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. In *AISTATS*, volume 15 of *JMLR Proceedings*, pages 208–214. JMLR.org, 2011.

Radu Ciucanu, Pascal Lafourcade, Marius Lombard-Platet, and Marta Soare. Secure best arm identification in multi-armed bandits. In *International Conference on Information Security Practice and Experience*, pages 152–171. Springer, 2019.

Radu Ciucanu, Anatole Delabrouille, Pascal Lafourcade, and Marta Soare. Secure cumulative reward maximization in linear stochastic bandits. In *International Conference on Provable Security*, pages 257–277. Springer, 2020.

Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume

7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

Sanchari Das, Robert S. Gutzwiller, Rod D. Roscoe, Prashanth Rajivan, Yang Wang, L. Jean Camp, and Roberto Hoyle. Panel: Humans and technology for inclusive privacy and security, 2021.

Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

Kefan Dong, Yingkai Li, Qin Zhang, and Yuan Zhou. Multinomial logit bandit with low switching cost. In *International Conference on Machine Learning*, pages 2607–2615. PMLR, 2020.

Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.

Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

Pedro M Esperança, Louis JM Aslett, and Chris C Holmes. Encrypted accelerated least squares regression. *arXiv preprint arXiv:1703.00839*, 2017.

Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption.

Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.

Benjamin Graham. Fractional max-pooling, 2015.

Chun-Hua Guo and Nicholas J Higham. A schur–newton method for the matrix\boldmath p th root and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 28(3):788–804, 2006.

Shai Halevi. Homomorphic encryption. In *Tutorials on the Foundations of Cryptography*, pages 219–276. Springer, 2017.

Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data.

Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose Blanchet, Peter W Glynn, and Yinyu Ye. Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321*, 2020.

Awni Y. Hannun, Brian Knott, Shubho Sengupta, and Laurens van der Maaten. Privacy-preserving contextual bandits. *CoRR*, abs/1910.05299, 2019.

Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991. doi: 10.1017/CBO9780511840371.

Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1209–1222, 2018.

Erich Kaltofen and Gilles Villard. On the complexity of computing determinants. *computational complexity*, 13(3-4):91–130, 2005.

Chamara Kattadige, Aravindh Raman, Kanchana Thilakarathna, Andra Lutu, and Diego Perino. 360norvic: 360-degree video classification from mobile encrypted video traffic. *arXiv preprint arXiv:2105.03611*, 2021.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013a.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 35–54. Springer, 2013b.

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

Vianney Perchet, Philippe Rigollet, Sylvain Chassang, Erik Snowberg, et al. Batched bandit problems. *The Annals of Statistics*, 44(2):660–681, 2016.

Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2):120–126, 1978.

Yufei Ruan, Jiaqi Yang, and Yuan Zhou. Linear bandits with limited adaptivity and learning distributional optimal design. *arXiv preprint arXiv:2007.01980*, 2020.

Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64. IEEE, 2015.

Neela Sawant, Chitti Babu Namballa, Narayanan Sadagopan, and Houssam Nassif. Contextual multi-armed bandits for causal marketing. *arXiv preprint arXiv:1810.01859*, 2018.

Abraham Seidenberg. Constructions in a polynomial ring over the ring of integers. *American Journal of Mathematics*, 100(4):685–703, 1978.

Roshan Shariff and Or Sheffet. Differentially private contextual linear bandits. In *NeurIPS*, pages 4301–4311, 2018.

Aristide C. Y. Tossou and Christos Dimitrakakis. Algorithms for differentially private multi-armed bandits. In *AAAI*, pages 2087–2093. AAAI Press, 2016.

Huazheng Wang, Qian Zhao, Qingyun Wu, Shubham Chopra, Abhinav Khaitan, and Hongning Wang. Global and local differential privacy for collaborative bandits. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, page 150–159. Association for Computing Machinery, 2020. ISBN 9781450375832.

X. Zhao and Ailan Wang. Generalized bootstrapping technique based on block equality test algorithm. *Secur. Commun. Networks*, 2018:9325082:1–9325082:8, 2018.

Kai Zheng, Tianle Cai, Weiran Huang, Zhenguo Li, and Liwei Wang. Locally differentially private (contextual) bandits learning. In *NeurIPS*, 2020.

Zhaowei Zhu, Jingxuan Zhu, Ji Liu, and Yang Liu. Federated bandit: A gossiping approach. *CoRR*, abs/2010.12763, 2020.

Supplementary Material: Encrypted Linear Contextual Bandit

A SLOW-SWITCHING ALGORITHM

In this section, we present the detailed algorithm of Sec. 4.

Algorithm 4 Low-Switching HELBA (Server-Side)

```
**Input:** horizon: $T$, regularization factor: $\lambda$, failure probability: $\delta$, feature bound: $L$, $\theta^\star$ norm bound: $S$, dimension: $d$, batch growth: $\eta$, trace condition: $C$
Set $w_1 = Enc_{\mathrm{pk}}(0)$, $\Lambda_1 = Enc_{\mathbf{pk}}(\lambda I)$, $\bar{A}_1 = Enc_{\mathbf{pk}}(\lambda^{-1} I)$, $\check{V}_1 = Enc_{\mathbf{pk}}(\lambda I)$ $\check{g}_0 = 0$, $j = 0$ and $t_0 = 1$
for $t = 1 \dots, T$ do
  Set $\widetilde{\beta}(t) = \sigma\sqrt{d \ln\left(\left(1 + \frac{L^2 t_j}{\lambda}\right)\left(\frac{\pi^2 t^2}{6\delta}\right)\right)} + t_j^{-1/2} + S\sqrt{\lambda}$ and $\epsilon_j = L(t_j^{3/2}\sqrt{\lambda + L^2 t_j})^{-1}$
  Observe encrypted contexts $(x_{t,a})_{a\in[K]} = (Enc_{\mathbf{pk}}(s_{t,a}))_{a\in[K]}$
  for $a = 1, \dots, K$ do
    Compute approximate square root $\mathrm{sqrt}_{\mathrm{HE}}\left(x_{t,a}^\top \bar{A}_j x_{t,a} + \varepsilon_j\right)$
    Compute encrypted indexes $\rho_a(t) = \langle x_{t,a}, w_j\rangle + \widetilde{\beta}(t)\left(\mathrm{sqrt}_{\mathrm{HE}}\left(x_{t,a}^\top \bar{A}_j x_{t,a} + \varepsilon_j\right) + t^{-1}\right)$ (Step ❷)
    Rescale encrypted indexes $\widehat{\rho}_a(t) = \frac{\rho_a(t) - r_{\min}}{\rho_{\max} - r_{\min}}$ with $\rho_{\max} = r_{\max} + 2\widetilde{\beta}(t)\left[\frac{2}{t} + \frac{L}{t^{3/2}\sqrt{\lambda + L^2 t}} + L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)\right]$
  end for
  Compute comparison vector $b_t \in \mathbb{R}^K$ using **acomp** (see Alg. 9 in App. D.5) with precision $\varepsilon'_t = (4.1t)^{-1}$ (Step ❸)
  Observe encrypted reward $y_t$ and encrypted context $x_{t,a_t}$
  Update $\check{V}_{t+1} = \check{V}_t + x_{t,a_t} x_{t,a_t}^\intercal$ and $\check{g}_{t+1} = \check{g}_t + y_t x_{t,a_t}$
  Compute Cond. (8) by computing $\delta_t$ with $\varepsilon = 0.45$ and $\varepsilon'_t = L^2(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}})(t - 1 - t_j))$ (see App. E.1.1).
  Use masking procedure on $\delta_t$ (Alg. 10) and sends the masked ciphertext to the user
  if $\delta_t \geq 0.45$ or $t \geq (1+\eta)t_j$ then
    Set $t_{j+1} = t$, $j = j+1$ and $\Lambda_{j+1} = \check{V}_t$
    Compute $\bar{A}_{j+1} = X_{k_1(\varepsilon_{j+1}/L^2)}$ as in Prop. 2 ($V = \Lambda_{j+1}$, $c = \lambda d + L^2 t_{j+1}$) and $w_{j+1} = \bar{A}_{j+1}\check{g}_{t_{j+1}}$
  end if
end for
```

B ADDITIONAL RELATED WORK

In Federated Learning (a.k.a., collaborative multi-agent), DP and LDP guarantees can provide a higher level of privacy at a small regret cost, leveraging collaboration between users Wang et al. (2020); Zhu et al. (2020). Another collaborative approach to privacy-preserving machine learning, called Secure Multi-Party Computation (MPC) (e.g. Damgård et al., 2012), divides computations between parties, while guarantying that it is not possible for any of them to learn anything about the others. This has been recently empirically investigated in the bandit framework Hannun et al. (2019). However, there is an additional strong assumption, that each party provides a subset of the features observed at each round.

Finally, Homomorphic Encryption (HE) (e.g. Halevi, 2017) aims at providing a set of tools to perform computation on encrypted data, outsourcing computations to potentially untrusted third parties (in our setting the bandit algorithm) since data cannot be decrypted. HE has only been merely used to encrypt rewards in bandit problems Ciucanu et al. (2020, 2019), but in some inherently simpler setting: i) contexts are not considered and arms' features are not encrypted; ii) a trusted party decrypts data. In particular, the second point makes algorithm design much easier but requires users to trust the third party which, in turn, can lead again to privacy/security concerns. In the supervised learning literature, HE has been used to train neural networks (Badawi et al., 2020) achieving 77.55% classification accuracy on CIFAR-10 (compared to a state-of-the-art accuracy of 96.53% (Graham, 2015)) highlighting the potentially high impact of the approximation error due to HE.

C PROTOCOL DETAILS

The learning algorithm may try to break encryption by inferring a mapping between ciphertexts and values or by storing all data. HE relies on the hardness of the *Learning With Error* problem (Albrecht et al., 2015) to guarantee security. To break an HE scheme, an attacker has to perform at least 2^κ operations to be able to differentiate noise from messages in a given ciphertext. We refer to (Albrecht et al., 2018) for a survey on the actual number of operations needed to break HE schemes with most of the known attacks. Although collecting multiple ciphertexts may speedup some attacks, the security of any HE scheme is still guaranteed as long as long the number of ciphertexts observed by an attacker is polynomial in N (Regev, 2009).

C.1 CKKS Encryption Scheme

In this section, we introduce the CKKS scheme Cheon et al. (2017). This scheme is inspired by the BGV scheme Brakerski et al. (2014) but has been modified to handle the encryption of real numbers. The security of those schemes relies on the assumption of hardness of the Learning With Errors (LWE), ring-LWE (RLWE) Regev (2009). The scheme can be divided into 2 parts: encoding/decoding and encryption/decryption.

C.1.1 Encoding and Decoding of Messages.

In CKKS, the space of message is defined as $\mathbb{C}^{N/2}$ for some big even integer $N \in \mathbb{N}$. This integer is a parameter of the scheme chosen when generating the private and secret keys. CKKS scheme does not work directly on the space $\mathbb{C}^{N/2}$ but rather on an integer polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\left(X^N + 1\right)$ (the plaintext space) Seidenberg (1978). Encoding a message $m \in \mathbb{C}^{N/2}$ into the plaintext space $\mathcal{R}$ is not as straightforward as using a classical embedding of a vector into a polynomial because we need the coefficients of the resulting polynomial to be integers. To solve this issue the CKKS scheme use a more sophisticated construction that the canonical embedding, based on the subring $\mathbb{H} = \{z \in \mathbb{C}^N \mid z_j = \bar{z}_{N-j}, j \leq N/2\}$ which is isomomorphic to $\mathbb{C}^{N/2}$. Finally, using a canonical embedding $\sigma : \mathcal{R} \to \sigma(\mathcal{R}) \subset \mathbb{H}$ and the *coordinate-wise random rounding* technique developed in Lyubashevsky et al. (2013b), the CKKS scheme is able to construct an isomorphism between $\mathbb{C}^{N/2}$ and $\mathcal{R}$.

C.1.2 Encryption and Decryption of Ciphertexts.

Most public key scheme relies on the hardness of the *Learning with Error* (LWE) problem introduced in Regev (2009). The LWE problem consists in distinguishing between noisy pairs $(a_i, \langle a_i, s\rangle + e_i)_{i\leq n} \subset (\mathbb{Z}/q\mathbb{Z})^n \times \mathbb{Z}/q\mathbb{Z}$ and uniformly sampled pairs in $(\mathbb{Z}/q\mathbb{Z})^n \times \mathbb{Z}/q\mathbb{Z}$ where $(e_i)_{i\leq n}$ are random noises and $q \in \mathbb{N}$. However, building a cryptographic public key system based on LWE is computationally inefficient. That's why CKKS relies on the *Ring Learning with Error* (RLWE) introduced in Lyubashevsky et al. (2013a) which is based on the same idea as LWE but working with polynomials $\mathbb{Z}_q[X]/(X^N + 1)$ instead on integer in $\mathbb{Z}/q\mathbb{Z}$. RLWE (and LWE) problem are assumed to be difficult to solve and are thus used as bases for cryptographic system. The security of those problems can be evaluated thanks to Albrecht et al. (2015) which gives practical bounds on the number of operations needed for known attacks to solve the LWE (RLWE) problem.

The CKKS scheme samples a random s on $\mathcal{R}$ and defines the secret key as $\mathbf{sk} = (1, s)$. It then samples a vector a uniformly on $\mathcal{R}/q_L\mathcal{R}$ (with $q_L = 2^L q_0$ where L is the depth of the scheme and q_0 its modulus) and an error term e sampled on $\mathcal{R}$ (usually each coefficient is drawn from a discrete Gaussian distribution). The public key is then defined as $\mathbf{pk} = (a, -a.s + e)$. Finally, to encrypt a message $m \in \mathbb{C}^{N/2}$ identified by a plaintext $\mathfrak{m} \in \mathcal{R}$ the scheme samples an encrypting noise $\nu \sim \mathcal{ZO}(0.5)$[4]. The scheme then samples $e_0, e_1 \in \mathbb{Z}^N$ two independent random variable from any distribution on $\mathcal{R}$, usually a discrete Gaussian distribution. The ciphertext associated to the message m is then $[(\nu \cdot \mathbf{pk} + (\mathfrak{m} + e_0, e_1))]_{q_L}$ with $[.]_{q_L}$ the modulo operator and $q_L = 2^L$. Finally, to decrypt a ciphertext $c = (c_0, c_1) \in \mathcal{R}^2_{q_l}$ (with l the level of the ciphertext, that is to say the depth of the ciphertext), the scheme computes the plaintext $\mathfrak{m}' = [c_0 + c_1 s]_{q_l}$[5] and returns the message m' associated to the plaintext $\mathfrak{m}'$.

[4]A random variable $X \sim \mathcal{ZO}(0.5)$ such that $X \in \{0, 1, -1\}^N$, $(X_i)_{i\leq N}$ are i.i.d such that for all $i \leq N$ $\mathbb{P}(X_i = 0) = 1/2$,$\mathbb{P}(X_i = 1) = 1/4$ and $\mathbb{P}(X_i = -1) = 1/4$

[5]for any $n \in \mathbb{N}$, $[.]_n$ is the remainder of the division by n

C.2 Key Switching

Homomorphic Encryption schemes needs all ciphertexts to be encrypted under the same public key in order to perform additions and multiplications. As we mentioned in Sec. 6 one way to circumvent this issue is to use a *KeySwitching* operation. The *KeySwitching* operation takes as input a cyphertext c_1 encrypted thanks to a public key pk_1 associated to a secret key sk_1 and transform it into a cyphertext encrypting the same message as c_1 but under a different secret key sk_2.

The exact *KeySwitching* procedure for each scheme is different. We will use the CKKS scheme, inspired by the BGV scheme Brakerski et al. (2014), where *KeySwitching* relies on two operations BitDecomp and PowerOf2, described below,

1. $\mathrm{BitDecomp}(c, q)$ takes as input a ciphertext $c \in \mathbb{R}^N$ with m the size of the ring dimension used in CKKS and an integer q. This algorithm decomposes c in its bit representation $(u_0, \dots, u_{\lceil \log_2(q) \rceil}) \in \mathbb{R}^{N \times \lceil \log_2(q) \rceil}$ such that $c = \sum_{j=0}^{\lfloor \log_2(q) \rfloor} 2^j u_j$

2. $\mathrm{PowerOf2}(c, q)$ takes as input a ciphertext $c \in \mathbb{R}^N$ and an integer q. This algorithm outputs $(c, 2c, \dots, 2^{\lfloor \log_2(q) \rfloor} c) \in \mathbb{R}^{m \times \lceil \log_2(q) \rceil}$

The *KeySwitching* operation can then be decomposed as:

- the first party responsible for sk_1 generates a new (bigger, in the sense that the parameter N is bigger than sk_1) public key $\tilde{\mathrm{pk}}_1$ still associated to sk_1

- the owner of secret key sk_2 computes $\mathrm{PowerOf2}(\mathrm{sk}_2)$ and add it to $\tilde{\mathrm{pk}}_1$. This object is called the *KeySwitchingKey*.

- the new cyphertext is computed by mulitiplying $\mathrm{BitDecomp}(c_1)$ with the KeySwitchingKey. This gives a new cyphertext decryptable with the secret key sk_2 and encrypted under a new public key pk_2

Algorithm 5 KeySwitching Procedure

```
**Input:** Cyphertext: $c$, User: $u$, User public key/secret key: $\mathrm{pk}_u, \mathrm{sk}_u$, Bandit Algorithm: $\mathfrak{A}$, Trusted Third Party: $\mathfrak{B}$, integer $q$
Alg. $\mathfrak{A}$ receives cypthertext $c$ encrpyted with key $\mathrm{pk}_u$
$\mathfrak{B}$ sends public key $\mathbf{pk}$ to $u$
$u$ computes $\mathrm{Enc}_{\mathbf{pk}_u}(\mathbf{ksk}_u) = \mathrm{Enc}_{\mathbf{pk}_u}(\mathrm{PowerOf2}(\mathbf{sk}_u, q) + \mathbf{pk})$
$u$ sends $\mathrm{Enc}_{\mathbf{pk}_u}(\mathbf{ksk}_u)$ to $\mathfrak{A}$
$\mathfrak{A}$ computes the new cyphertext $c' = \mathrm{Enc}_{\mathbf{pk}_u}(\mathrm{BitDecomp}(c, q)^\intercal)\mathrm{Enc}_{\mathbf{pk}_u}(\mathbf{ksk}_u) = \mathrm{Enc}_{\mathbf{pk}_u}(\mathrm{Enc}_{\mathbf{pk}}(c))$
$u$ decrypts $c'$ and sends the result to $\mathfrak{A}$
```

Alg. 5 allows us to perform the *KeySwitching* in a private manner for the CKKS scheme. Indeed, the key switch operation requires to decompose a secret key thanks to the PowerOf2 procedure. If not done in a secure fashion this could lead to a leak of the frist private key. It is thus necessary to ensure that this key is not distributed in the clear. However, our private procedure requires communication between the bandit algorithm $\mathfrak{A}$ and the user u. In particular, the user still needs to receives the public key from the trusted third party. However, the user does not need to be known ahead of time as previously.

D TOWARD AN ENCRYPTED OFUL

In this section, we provide the proof of the results of Step ❶, ❷ and ❸, i.e., the speed of convergence of iterating Eq. (2) or Eq. (5), how to build a confidence intervals around $\theta^\star$ and how the approximate argmax is computed in Alg. 3.

D.1 Computing an Approximate Inverse

First, we prove Prop. 2. The proof of convergence the Newton method for matrix inversion is rather standard but the proof of convergence for the stable method (Eq. (2)) is often not stated. We derive it here for completeness. First, we recall Prop. 2.

Proposition. *Given a symmetric positive definite matrix $V \in \mathbb{R}^{d\times d}$, $c \geq Tr(V)$ and a precision level $\varepsilon > 0$, the iterate in* (2) *satisfies*

$$\|X_k - V^{-1}\| \leq \varepsilon$$

for any $k \geq k_1(\varepsilon)$ with

$$k_1(\varepsilon) = \frac{1}{\ln(2)} \ln\left(\frac{\ln(\lambda) + \ln(\varepsilon)}{\ln\left(1 - \frac{\lambda}{c}\right)}\right)$$

, where $\lambda \leq \lambda_d$ is a lower bound to the minimal eigenvalue of V and $\|\cdot\|$ is the matrix spectral-norm.

Proof. of Prop. 2. After k iterations of Eq. (2), we have that $VX_k = M_k$. Indeed we proceed by induction:

- For $k = 0$, $M_0 = \frac{1}{c}V = VX_0$
- For $k+1$ given the property at time k, $VX_{k+1} = VX_k(2I_d - M_k) = M_k(2I_d - M_k) = M_{k+1}$

Let's note $E_k = X_k - V^{-1}$ and $\tilde{E}_k = M_k - I_d$ then:

$$\begin{aligned} E_{k+1} = (X_{k+1}V - I_d)\,V^{-1} &= (M_{k+1} - I_d)\,V^{-1} \\ &= -\left(M_k^2 - 2M_k + I_d\right)V^{-1} \\ &= -\left(M_k - I_d\right)^2 V^{-1} = -\tilde{E}_k^2 V^{-1} \end{aligned}$$

where the second equality is possible because V and $(X_k)_{k\in\mathbb{N}}$ commute as for all $k \in \mathbb{N}$, X_k is a polynomial function of V.

Therefore, we have for any $k \in \mathbb{N}$:

$$\|E_{k+1}\| = \|\tilde{E}_k^2 V^{-1}\| \leq \|V^{-1}\| \times \|\tilde{E}_k\|^2 \tag{10}$$

But at the same time:

$$\|\tilde{E}_{k+1}\| = \|M_{k+1} - I_d\| = \|M_k(2I_d - M_k) - I_d\| = \|-(M_k - I_d)^2\| \leq \|\tilde{E}_k\|^2 \tag{11}$$

thus iterating Eq. (11), we have that for all $k \in \mathbb{N}$, $\|\tilde{E}_k\| \leq \|\tilde{E}_0\|^{2^k}$. And then $\|\tilde{E}_k\| \leq \|\tilde{E}_0\|^{2^k}\|V^{-1}\|$, therefore using that any V symmetric definite positive $\|V^{-1}\| = \|V\|^{-1}$ then for all $k \in \mathbb{N}$:

$$\|E_k\| \leq \left\|\frac{V}{c} - I_d\right\|^{2^k} \|V\|^{-1} \tag{12}$$

But $\|\tilde{E}_0\| = \left\|\frac{1}{c}V - I_d\right\| = \max_{i\in[d]}\left|\frac{\lambda_i}{c} - 1\right|$ where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$ are the (ordered) eigenvalues of V. However $c \geq \mathrm{Tr}(V)$ thus $0 \leq \lambda_i/c \leq 1$ for all $i \leq d$. Therefore $\|\tilde{E}_0\| \leq 1 - \frac{\lambda_d}{c}$. We also have that $\|V\| = \lambda_1$. Using Eq. (12), we have for all k:

$$\|E_k\| \leq \left(1 - \frac{\lambda_d}{c}\right)^{2^k} \lambda_1^{-1} \leq \left(1 - \frac{\lambda}{c}\right)^{2^k} \lambda^{-1} \tag{13}$$

for any $0 \geq \lambda \leq \lambda_d$. Finally, Eq. (13) implies that $\|E_k\| \leq \varepsilon$ as soon as:

$$k \geq \frac{1}{\ln(2)} \ln\left(\frac{\ln(\lambda) + \ln(\varepsilon)}{\ln\left(1 - \frac{\lambda}{c}\right)}\right) \tag{14}$$

for any $0 \geq \lambda \leq \lambda_d$ and $\lambda\varepsilon \leq 1$. □

D.2 Computing an Approximate Square Root

The proof of Prop. 4 is very similar to the proof of Prop. 2 thanks the analysis of the convergence speed in Cheon et al. (2019). First, let us recall Prop. 4.

Proposition. *For any $z \in \mathbb{R}_+$, $c_1, c_2 > 0$ with $c_2 \geq z \geq c_1$ and a precision $\varepsilon > 0$, let q_k be the result of k iterations of Eq. (5), with $q_0 = \frac{z}{c_2}$ and $v_0 = \frac{z}{c_2} - 1$. Then, $|q_k\sqrt{c_2} - \sqrt{z}| \leq \varepsilon$ for any $k \geq k_0(\varepsilon) := \frac{1}{\ln(2)} \ln\left(\frac{\ln(\varepsilon)-\ln(\sqrt{c_2})}{4\ln\left(1-\frac{c_1}{4c_2}\right)}\right)$.*

Proof. of Prop. 4. Because $0 \leq c_1 < x < c_2$, we have that $\frac{x}{c_2} \in (0,1)$, hence thanks to Lemma 2 of Cheon et al. (2019), we have that after k iterations:

$$\left|q_k - \sqrt{\frac{x}{c_2}}\right| \leq \left(1 - \frac{x}{4c_2}\right)^{2^{k+1}} \tag{15}$$

where q_k is the k-th iterate from iterating Eq. (5) with $q_0 = \frac{x}{c_2}$ and $v_0 = q_0 - 1$. Then because $x \geq c_1$, we have that $1 - \frac{x}{4c_2} \leq 1 - \frac{c_1}{4c_2}$. Stated otherwise,

$$\left|q_k - \sqrt{\frac{x}{c_2}}\right| \leq \left(1 - \frac{c_1}{4c_2}\right)^{2^{k+1}} \tag{16}$$

Therefore, for $k \geq \frac{1}{\ln(2)} \ln\left(\frac{\ln(\varepsilon)-\ln(\sqrt{c_2})}{2\ln\left(1-\frac{c_1}{4c_2}\right)}\right)$, the result follows since:

$$\sqrt{c_2}\left|q_k - \sqrt{\frac{x}{c_2}}\right| \leq \varepsilon \tag{17}$$

□

D.3 Computing an Optimistic Ellipsoid Width.

The next step to build an optimistic algorithm is to compute a confidence ellipsoid around the estimate $\widetilde{\theta}_t$ such that the true parameter $\theta^\star$ belongs to this confidence ellipsoid with high probability. First, we need an estimate of the distance between $\theta^\star$ and $\widetilde{\theta}_t$ that is the object of Cor. 3. The proof of Cor. 3, is based on the fact that the approximated inverse is closed enough to the true inverse. Let's recall Cor. 3 first.

Corollary. *Setting $\varepsilon_t = \left(Lt^{3/2}\sqrt{L^2t+\lambda}\right)^{-1}$ in Prop. 2, then $\|Dec_{\mathbf{sk}}(\omega_t) - \theta_t\|_{V_t} \leq t^{-1/2}$, $\forall t$.*

Proof. of Cor. 3. Let's note $\bar{A}_t$, the result of iterating Eq. (2), $k_1(\varepsilon_t)$ times with $V = V_t$ and $c = \lambda d + L^2 t$. Thanks to the definition of $\mathrm{Dec}_{\mathbf{sk}}(w_t)$ and $\theta_t = V_t^{-1}b_t$, we have:

$$\begin{aligned}
\|\mathrm{Dec}_{\mathbf{sk}}(w_t) - \theta_t\|_{V_t} &= \left\|V_t^{1/2}\left(V_t^{-1} - \mathrm{Dec}_{\mathbf{sk}}(\bar{A}_t)\right)\sum_{l=1}^{t-1} r_l s_{l,a_l}\right\|_2 \qquad (18)\\
&= \left\|\left(V_t^{-1} - \mathrm{Dec}_{\mathbf{sk}}(\bar{A}_t)\right)V_t^{1/2}\sum_{l=1}^{t-1} r_l s_{l,a_l}\right\|_2 \qquad (19)\\
&\leq \|\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_t) - V_t^{-1}\| \left\|V_t^{1/2}\sum_{l=1}^{t-1} r_l s_{l,a_l}\right\|_2 \qquad (20)
\end{aligned}$$

But $\mathrm{Tr}(V_t) \leq \lambda d + L^2 t$ and $\lambda_{\min}(V_t) \geq \lambda$. Therefore thanks to Prop. 2 $\bar{A}_t$ is such that:

$$\|\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_t) - V_t^{-1}\| \leq \varepsilon_t \tag{21}$$

We also have that:

$$\begin{aligned}
\left\|V_t^{1/2}\sum_{l=1}^{t-1} r_l s_{l,a_l}\right\|_2 &\leq \|\sqrt{V_t}\| \left\|\sum_{l=1}^{t-1} r_l s_{l,a_l}\right\|_2 \qquad (22)\\
&\leq Lt\sqrt{\|V_t\|} \leq Lt\sqrt{\lambda + L^2 t} \qquad (23)
\end{aligned}$$

because $r_l \in [-1,1]$ for all $l \leq t$ and $\lambda_{\max}(V_t) \leq \lambda + L^2 t$. Finally, we have that:

$$\|\theta_t - \tilde{\theta}_t\|_{V_t} \leq \varepsilon_t L t \sqrt{\lambda + L^2 t} \leq t^{-1/2} \tag{24}$$

□

D.4 Approximate Confidence Ellipsoid

Finally thanks to Cor. 3, we can now prove that with high probability $\theta^\star$ belongs to the inflated confidence intervals $\tilde{\mathcal{C}}_t$ for all time t. That is the object of Prop. 9.

Proposition 9. *For any $\delta > 0$, we have that with probability at least $1-\delta$:*

$$\theta^\star \in \bigcap_{t=1}^{+\infty} \mathcal{C}_t(\delta) := \left\{ \theta \mid \|\theta - Dec_{\boldsymbol{sk}}(w_t)\|_{V_t} \leq \widetilde{\beta}(t) \right\} \tag{25}$$

with $\widetilde{\beta}(t) = t^{-1/2} + \sqrt{\lambda} S + \sigma\sqrt{d(\ln(1 + L^2 t/(\lambda d)) + \ln(\pi^2 t^2/(6\delta))}$

Proof. of Prop. 9. Using Cor. 3 and Thm. 2 in Abbasi-Yadkori et al. (2011), we have that for any time t that with probability at least $1-\delta$:

$$\begin{aligned}\|\theta^\star - \mathrm{Dec}_{\mathbf{sk}}(w_t)\|_{V_t} &\leq \|\theta_t - \mathrm{Dec}_{\mathbf{sk}}(w_t)\|_{V_t} + \|\theta^\star - \theta_t\|_{V_t} &(26)\\ &\leq t^{-1/2} + \sqrt{\lambda} S + \sigma\sqrt{d(\ln(1 + L^2 t/(\lambda d)) + \ln(1/\delta))} &(27)\end{aligned}$$

where w_t computed as in Alg. 4 and θ_t is the ridge regression estimate computed at every time step in OFUL. Taking a union bound with high-probability event means that with probability at least $1 - \frac{6\delta}{\pi^2}$, we have:

$$\begin{aligned}\|\theta^\star - \theta_t\|_{V_t} &\leq \|\theta_t - \mathrm{Dec}_{\mathbf{sk}}(w_t)\|_{V_t} + \|\theta^\star - \theta_t\|_{V_t} &(28)\\ &\leq t^{-1/2} + \sqrt{\lambda} S + \sigma\sqrt{d(\ln(1 + L^2 t/(\lambda d)) + \ln(\pi^2 t^2/(6\delta)))} &(29)\end{aligned}$$

□

D.5 Homomorphic Friendly Approximate Argmax

As mentioned in Sec. 4, an homomorphic algorithm can not directly compute the argmax of a given list of values. In this work, we introduce the algorithm Alg. 9 to compute the comparison vector $b_t \approxeq \left(\mathbb{1}_{\{a = \arg\max_{i\in[K]} \rho_i(t)\}}\right)$ with $(\rho_a(t))_{a\in[K]}$ the UCBs defined in Sec. 4. This algorithm is divided in two parts. First, it computes an approximate maximum, M of $(\rho_a(t))_{a\in[K]}$ thanks to Alg. 8 and then compares each values $(\rho_a(t))_{a\in[K]}$ to this approximate maximum M thanks to the algorithm **NewComp** of Cheon et al. (2020) (recalled as Alg. 6).

Algorithm 6 NewComp

```
Input: Entry numbers: $a, b \in [0,1]$, $n$ and depth $d$
Set $x = a - b$
for $k = 1, \dots, d$ do
    Compute $x = f_n(x) = \sum_{i=0}^{n} \frac{1}{4^i}\binom{2i}{i} x(1-x^2)^i$
end for
Return: $(x+1)/2$
```

Rescaling the UCB index: In order to use the HE-friendly algorithms of Cheon et al. (2020), we need to rescale the UCB-index to lies in $[0,1]$. Determining the range of those indexes is the purpose of the following proposition.

Proposition 10. *For every time $t \geq 1$, assuming $r_l \in [-1,1]$ for any $l \leq t$ and $L \geq 1$ then for any $\delta > 0$ we have that with probability at least $1-\delta$:*

$$-1 \leq \rho_a(t) \leq 1 + 2\widetilde{\beta}(t)\left[2t^{-1} + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}} + \sqrt{\frac{L}{t^{3/2}\sqrt{\lambda + L^2 t}}}\right] \tag{30}$$

where $\rho_a(t) = \langle \widetilde{\theta}_t, x_{t,a}\rangle + \widetilde{\beta}(t)\left[q_{k_0(t^{-1})} + t^{-1}\right]$ *the UCB index of arm a at time t.*

Algorithm 7 NewMax

```
**Input:** Entry numbers: $a, b \in [0,1]$, $n$ and depth $d$
Set $x = a - b$, $y = \frac{a+b}{2}$
**for** $k = 1, \dots, d$ **do**
    Compute $x = f_n(x) = \sum_{i=0}^{n} \frac{1}{4^i}\binom{2i}{i} x(1-x^2)^i$
**end for**
**Return:** $y + \frac{a+b}{2} \cdot x$
```

Algorithm 8 amax

```
**Input:** Entry numbers: $(a_i)_{i \le K}$, $n$ and depth $d$
Set $m = a_1$
**for** $i = 2, \dots, K$ **do**
    Compute $m = \max\{m, a_i\}$ thanks to **NewMax** in Cheon et al. (2020) with parameter $a = m$, $b = a_i$, $n$ and $d$
**end for**
```

Proof. of Prop. 10. For $\delta > 0$, we denote $E = \bigcap_{l=1}^{+\infty} \left\{\theta^\star \in \tilde{\mathcal{C}}_l(\delta)\right\}$ so that, using Prop. 9, $\mathbb{P}(E) \ge 1 - \delta$. Under the event E, we have for any arm a:

$$-1 \le \langle x_{t,a}, \theta^\star \rangle \le \rho_a(t) \le \langle x_{t,a}, \theta^\star \rangle + 2\widetilde{\beta}(t)\left[q_{k_0(1/t)} + t^{-1}\right] \tag{31}$$

On the other hand thanks to Prop. 4, we have that $q_{k_0(t^{-1})} \le \sqrt{\|x\|^2_{\mathrm{Dec}_{\mathbf{sk}}(A_t)} + \frac{L}{t^{3/2}\sqrt{\lambda + L^2 t}}} + t^{-1}$. and also $\|x\|^2_{A_t} \le L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)$.

Indeed because $\mathrm{Dec}_{\mathbf{sk}}(A_t)$ is a polynomial function of V_t, we have that $\mathrm{Dec}_{\mathbf{sk}}(A_t)$ is symmetric and $\mathrm{Dec}_{\mathbf{sk}}(A_t)V_t = V_t\mathrm{Dec}_{\mathbf{sk}}(A_t)$, hence $\mathrm{Dec}_{\mathbf{sk}}(A_t)$ and V_t^{-1} are diagonalizable in the same basis therefore $\|\mathrm{Dec}_{\mathbf{sk}}(A_t) - V_t^{-1}\| = \max_{i \le d} |\lambda_i(\mathrm{Dec}_{\mathbf{sk}}(A_t)) - \lambda_i(V_t^{-1})|$ with $\lambda_i(M)$ the i-th biggest eigenvalue of M. Hence:

$$\lambda_1(\mathrm{Dec}_{\mathbf{sk}}(A_t)) \le \frac{1}{\lambda} + \frac{1}{Lt^{3/2}\sqrt{\lambda + L^2 t}} \tag{32}$$

and:

$$\lambda_d(\mathrm{Dec}_{\mathbf{sk}}(A_t)) \ge \frac{1}{\lambda + L^2 t} - \frac{1}{Lt^{3/2}\sqrt{\lambda + L^2 t}} > 0 \tag{33}$$

for $t \ge 2$. Therefore, we have that for any arm a:

$$\rho_a(t) \le \langle \theta^\star, x_{t,a} \rangle + 2\widetilde{\beta}(t)\left[2t^{-1} + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}} + \sqrt{\frac{L}{t^{3/2}\sqrt{\lambda + L^2 t}}}\right] \tag{34}$$

□

Computing the Comparaison Vector: The algorithm Alg. 9 operates on values in $[0,1]$ therefore using Prop. 10, we can compute rescaled UCB index, noted $\tilde{\rho}_a(t) \in [0,1]$. We are then almost ready to prove Cor. 5, we just need two lemmas which relates the precision of Alg. 9 and Alg. 8 to the precision of **NewComp** and **NewMax** of Cheon et al. (2020).

The first lemma (Lem. 11) gives a lower bound on the depth needed for Alg. 8 to achieve a given precision.

Lemma 11. *For any sequences $(a_i)_{i \le K} \in [0,1]^K$, for any precision $0 < \varepsilon < K/4$, $n \in \mathbb{N}^\star$ and*

$$d(\varepsilon, n) \ge \frac{\ln\left(\frac{\ln\left(\frac{K}{\varepsilon}\right)}{\ln(2)} - 2\right)}{\ln(c_n)} \tag{35}$$

with $c_n = \frac{2n+1}{4^n}\binom{2n}{n}$. Noting M the result of Alg. 8 with parameter $(a_i)_i$, n and $d(\varepsilon, n)$, we have that:

$$\left|M - \max_i a_i\right| \le \varepsilon \tag{36}$$

Algorithm 9 acomp

```
Input: Entry numbers: $(a_i)_{i\le K}$, precision $\varepsilon$
Set depth $d = 1 + \left\lfloor 3.2 + \frac{\ln(1/\varepsilon)}{\ln(3/2)} + \frac{\ln\left(\frac{\ln(1/\varepsilon)}{\ln(2)}-2\right)}{\ln(2)} \right\rfloor$ and depthmax $d' = \frac{1}{\ln(3/2)} \ln\left(\frac{\alpha\ln\left(\frac{1}{\varepsilon}\right)}{\ln(2)} - 2\right)$ with $\alpha = \frac{3}{2} + \frac{5.2\ln(3/2)}{\ln(4)} + \frac{\ln(3/2)}{2\ln(2)}$
Compute $M = \text{amax}((a_i)_{i\le K}, n, d)$
for $i = 2, \ldots, K$ do
  Set $b_i = \textbf{NewComp}(a_i, M, n, d')$
end for
```

Proof. of Lemma 11. Thanks to Corollary 4 in Cheon et al. (2020), we have that for any n and depth $d \ge \frac{\ln\left(\frac{\ln(1/\varepsilon)}{\ln(2)}-2\right)}{\ln(c_n)}$ (with $c_n = \frac{2n+1}{4^n}\binom{2n}{n}$) and number a, b:

$$|\textbf{NewMax}(a,b,n,d) - \max\{a,b\}| \le \varepsilon \tag{37}$$

Let's note m_k the iterate m of Alg. 8 at step $k \in [K]$ in the for loop. We show that by induction $\left|m_k - \max_{i\in[k]} a_i\right| \le k\varepsilon$.

- By definition $m_1 = a_1$ and $|m_1 - \max_{i\le 1} a_i| = 0$
- Using that $|\max\{a,c\} - \max\{b,c\}| \le |a-b|$ for any $a,b,c \in \mathbb{R}$, we have:

$$\begin{aligned}\left|m_{k+1} - \max_{i\le k+1} a_i\right| &= \Big|\textbf{NewMax}(m_k, a_{k+1}, n, d) - \max\{m_k, a_{k+1}\} \\ &\qquad + \max\{m_k, a_{k+1}\} - \max\{\max_{i\le k} a_i, a_{k+1}\}\Big| \\ &\le |\textbf{NewMax}(m_k, a_{k+1}, n, d) - \max\{m_k, a_{k+1}\}| \\ &\qquad + |\max\{m_k, a_{k+1}\} - \max\{\max_{i\le k} a_i, a_{k+1}\}| \\ &\le \varepsilon + |m_k - \max_{i\le k} a_i| \le (k+1)\varepsilon\end{aligned}$$

Finally, because $M = m_K$, we just need to choose $d \ge \frac{\ln\left(\frac{\ln(K/\varepsilon)}{\ln(2)}-2\right)}{\ln(c_n)}$ to get the result. □

The next lemma (Lem. 12) has the same purpose of Lem. 11 but this time for Alg. 9. The proof is based on properties of the polynomial function used by the algorithm **NewComp** in order to predict the result of the comparison when the margin condition of **NewComp** (that is to say the result of the comparison of $a, b \in [0,1]$ is valid if and only if $|a-b| \ge \varepsilon$ for some $\varepsilon > 0$) is not satisfied.

Lemma 12. *For $\varepsilon \in (0, 1/4)$ and sequence $(a_i)_{i\le K} \in [0,1]^K$, let's denote $(b_i)_{i\le K}$ te result of Alg. 9 ruuned with parameter $(a_i)_{i\le K}$, $n = 1$, $d' = d_2(\varepsilon)$ and $d = d_3(\varepsilon)$ with:*

$$d_2(\varepsilon) = \left\lfloor 3.2 + \frac{\ln(1/\varepsilon)}{\ln(c_n)} + \frac{\ln\left(\ln\left(\frac{1}{\varepsilon}\right)/\ln(2) - 2\right)}{\ln(n+1)} \right\rfloor + 1 \tag{38}$$

$$d_3(\varepsilon) \ \ge \frac{1}{\ln(c_n)} \ln\left(\frac{\alpha\ln\left(\frac{1}{\varepsilon}\right)}{\ln(2)} - 2\right) \tag{39}$$

where $\alpha = \frac{3}{2} + \frac{5.2\ln(c_n)}{\ln(4)} + \frac{\ln(c_n)}{2\ln(n+1)}$. Then selecting any $i \le K$ such that $b_i \ge \varepsilon$ (and there is at least one such index i), we have that $a_i \ge \max_k a_k - 2\varepsilon$

Proof. of Lemma 12. Thanks to Corollary 1 in Cheon et al. (2019), we have that for each $i \le K$, $|b_i - \text{Comp}(a_i, M)| \le \varepsilon$ as soon as $|a_i - M| > \varepsilon$ and $d' = \left\lfloor 3.2 + \frac{\ln(1/\varepsilon)}{\ln(c_n)} + \frac{\ln\left(\ln\left(\frac{1}{\varepsilon}\right)/\ln(2)-2\right)}{\ln(n+1)} \right\rfloor + 1$. For $i \in [K]$, we have that:

- If $\max_{k\leq K} a_k \geq a_i \geq M+\varepsilon$ then $\mathrm{Comp}(a_i, M) = 1$, $|b_i - 1| \leq \varepsilon$ and $a_i \geq \max_{k\leq K} a_k - |\max_{k\leq K} a_k - M| - \varepsilon$
- If $a_i \leq M - \varepsilon$ then $\mathrm{Comp}(a_i, M) = 0$, thus $|b_i| \leq \varepsilon$ and $a_i \leq \max_{k\leq K} a_k + |\max_{k\leq K} a_k - M| - \varepsilon$

Therefore for any a_i such that $|a_i - M| > \varepsilon$ then the resulting b_i is either bounded by $1-\varepsilon$ or ε.

The second option is if $|a_i - M| \leq \varepsilon$ then the **NewComp** algorithm provides no guarantee to the result of the algorithm. However the algorithm applies a function f_n[6] multiple times to its input. For every $x \in [-1, 1]$:

$$|f_n(x)| \leq c_n|x| \text{ and } f_n([-1,1]) \subset [-1,1] \tag{40}$$

with $c_n = \frac{2n+1}{4^n}\binom{2n}{n}$. Hence:

$$\forall x \in [-1,1] \qquad |f_n^{(d')}(x)| \leq c_n |f_n^{(d'-1)}(x)| \leq c_n^{d"}|x| \tag{41}$$

But if $|a_i - M| \leq \varepsilon$, $f_n^{(d')}(a_i - M) \leq c_n^{d'}|a_i - M| \leq c_n^{d'}\varepsilon$ thus $\left|b_i - \frac{1}{2}\right| \leq \frac{c_n^{d'}\varepsilon}{2}$.

Finally for each i, we only three options for b_i:

- If $|a_i - M| \leq \varepsilon$ then $\left|b_i - \frac{1}{2}\right| \leq \frac{c_n^{d'}\varepsilon}{2}$ and $a_i \geq \max_k a_k \;\; - (\varepsilon + |\max_k a_k - M|) \geq \max_k a_k \;\; - 2\varepsilon$
- If $|a_i - M| \geq \varepsilon$ and $a_i \leq M - \varepsilon$ then $|b_i| \leq \varepsilon$ and $a_i \leq \max_k a_k + |M - \max_k a_k| - \varepsilon \leq \max_k a_k$
- If $|a_i - M| \geq \varepsilon$ and $a_i \geq M + \varepsilon$ then $|b_i - 1| \leq \varepsilon$ and $a_i \geq \max_k a_k - (|M - \max_k a_k| + \varepsilon) \geq \max_k a_k - 2\varepsilon$

To finish the proof, we just need to ensure that there exists at least one i such that $b_i \geq \varepsilon$. Noting $i^\star = \arg\max_k a_k$, if the amax algorithm is used with depth d such that:

$$d \geq \frac{1}{\ln(c_n)} \ln\left(\frac{\alpha \ln\left(\frac{1}{\varepsilon}\right)}{\ln(2)} - 2\right) \tag{42}$$

where $\alpha = \frac{3}{2} + \frac{5.2\ln(c_n)}{\ln(4)} + \frac{\ln(c_n)}{2\ln(n+1)}$, we have $|a_{i^\star} - M| \leq \varepsilon^\alpha \leq \varepsilon$ and $b_{i^\star} \geq \frac{1}{2} - \frac{c_n^{d'}\varepsilon^\alpha}{2} > \varepsilon$. Hence there always exists an index i such that $b_i \geq \varepsilon$. □

Finally, thanks to Lem. 12, we can finally prove Cor. 5. The proof of this corollary simply amounts to choose the right precision for **NewComp** algorithm at every step of Alg. 9. First let's recall Cor. 5.

Corollary. *For any time t, selecting any arm a such that $(b_t)_a \geq \frac{1}{4t}$ then:*

$$\rho_a(t) \geq \max_{k\leq K} \rho_k(t) - \frac{1}{t}\left(1 + \widetilde{\beta}^\star(t)\right) \tag{43}$$

where $\widetilde{\beta}^\star(t) = \widetilde{\beta}(t)\left[2t^{-1} + \sqrt{\frac{L}{t^{3/2}\sqrt{\lambda + L^2 t}}} + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}}\right]$

Proof. of Cor. 5. Using Lem. 12 with $\varepsilon = \frac{1}{4t}$ yields the following result:

$$\tilde{\rho}_i(t) \geq \max_{k\leq K} \tilde{\rho}_k(t) - \frac{1}{2t} \tag{44}$$

But for any $i \leq K$, $\tilde{\rho}_i(t) = (\rho_i(t)+1)/\left(2 + 2\widetilde{\beta}(t)\left[2t^{-1} + \sqrt{\frac{L}{t^{3/2}\sqrt{\lambda + L^2 t_j}}} + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}}\right]\right)$. Hence the result. □

E SLOW SWITCHING CONDITION AND REGRET OF HELBA

In this appendix, we present the analysis of the regret of HELBA. The proof is decomposed in two steps. The first one is the analysis of the number of batches for any time T. That is the object of the Sec. E.1. The second part of the proof amounts to bounding the regret as a function of the number of batches (Sec. E.2).

[6]For all $x \in [-1,1]$, $f_n(x) = \sum_{i=0}^{n} \frac{1}{4^i}\binom{2i}{i} x(1-x^2)^i$.

E.1 Number of batches of HELBA (Proof of Prop. 7)

We first prove Prop. 7 which states that the total number of batches for HELBA is logarithmic in T contrary to HELBA where the parameter are updated a linear number of times. The proof of this proposition is itself divided in multiple steps. First, we show how using **NewComp** to compare the parameter C and $\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)$ (for any batch j) relate to the comparison of C and $\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)$. Then, we show how Condition 8 relates to the det-based condition used in RSOFUL which allows us to finish the proof of Prop. 7 following the same reasoning as in Abbasi-Yadkori et al. (2011).

E.1.1 Homomorphically Friendly Comparison for Condition 8

We first prove the following proposition, bounding the error made by our algorithm when using $\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)$ instead of $\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)$.

Proposition 13. *For an batch j, time $t \geq t_j+1$, $\varepsilon < 1/2$ and $\varepsilon' > 0$, let's note δ_t the result of* ***NewComp*** *applied with parameters $a = \frac{Tr\left(Dec_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)}{L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)}$, $b = \frac{C}{L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)}$* [7]*, $n = 1$ and $d_5(\varepsilon)$ such that:*

$$d_5(\varepsilon) \geq 3.2 + \frac{\ln(1/\varepsilon')}{\ln(c_n)} + \frac{\ln\left(\ln\left(\frac{1}{\varepsilon}\right)/\ln(2) - 2\right)}{\ln(n+1)} \tag{45}$$

then:

- *if $\delta_t > \varepsilon$:*

$$C - \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j) \leq Tr\left(Dec_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \tag{46}$$

- *else if $\delta_t \leq \varepsilon$:*

$$Tr\left(Dec_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \leq C + \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j) \tag{47}$$

Proof. of Prop. 13. We consider the two cases, depending if δ_t is bigger than ε or not.

If $\delta_t > \varepsilon$: we proceed by separation of cases.

- If $\left|\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) - C\right| > \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$:

$$\left|\delta_t - \mathrm{Comp}\left(\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right), C\right)\right| \leq \varepsilon$$

  thanks to Cor. 1 in Cheon et al. (2020) for the precision of **NewComp**. We also used the fact that for any $x, y \in \mathbb{R}$ and $z \in \mathbb{R}_+^\star$, $\mathrm{Comp}(x/z, y/z) = \mathrm{Comp}(x, y)$. Using the equation above:

$$\mathrm{Comp}\left(\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right), C\right) \geq \delta_t - \varepsilon > 0$$

  because we assumed here that $\delta_t > \varepsilon$. This readily implies that $\mathrm{Comp}\left(\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right), C\right) = 1$ because $\mathrm{Comp}(a, b) \in \{0, 1\}$ for any $a, b \in [0, 1]$.

[7]with the convention that $0/0 = 0$ and $C/0 = 1$

But, because we are in the case that:

$$\left|\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) - C\right| > \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$$

we have that either $\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) > C + \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$ or $\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) < C - \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$. Hence, because $\operatorname{Comp}\left(\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right), C\right) = 1$, we have that $\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) > C$ that is to say:

$$\begin{aligned}\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) &\geq C + \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)\\ &\geq C - \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)\end{aligned}$$

- If $\left|\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) - C\right| \leq \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$: We can not use Cor. 5 from Cheon et al. (2020). However, in this case we directly have by definition of the absolute value that:

$$-\varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j) \leq \operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) - C \tag{48}$$

If $\delta_t \leq \varepsilon$: Again, we distinguish the two different cases possible.

- If $\left|\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) - C\right| > \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$: Using Cor. 1 from Cheon et al. (2020), we have once again that:

$$\left|\delta_t - \operatorname{Comp}\left(\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right), C\right)\right| \leq \varepsilon$$

Therefore $\operatorname{Comp}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}, C\right) \leq \delta_t + \varepsilon \leq 2\varepsilon < 1$ (because $\varepsilon < 1/2$). But $\operatorname{Comp}(a,b) \in \{0,1\}$ for any $a,b \in [0,1]$ which means that $\operatorname{Comp}\left(\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right), C\right) = 0$. But we assumed that $\left|\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) - C\right| > \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$, in other words:

$$\begin{aligned}&\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) > C + \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j) \geq C \text{ or}\\ &\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) < C - \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)\end{aligned} \tag{49}$$

But $\operatorname{Comp}\left(\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right), C\right) = 0$, it is thus only possible that

$$\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C - \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$$

- If $\left|\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) - C\right| \leq \varepsilon'$:
In this case, by definition we have

$$\operatorname{Tr}\left(\operatorname{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C + \varepsilon' L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)$$

□

The previous proposition ensures that when a batch is ended because $\delta_t > 0.45$ then we have, for a small enough ε', that, $\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} x_{l,a_l}x_{l,a_l}^\intercal\right) \geq C'$ for some constant C'. However, thanks to Prop. 2 we have that for any batch j, that for all $l \in \{t_j+1, \ldots, t-1\}$:

$$\|x\|^2_{\bar{V}_j^{-1}} - \|x\|^2_{\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)} \leq L^2\|\bar{V}_j^{-1} - \mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\| \leq \frac{L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \tag{50}$$

Summing over all time steps $l \in [t_j+1, t-1]$, we have that:

$$\left|\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) - \mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)\right| \leq \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \tag{51}$$

Therefore when $\delta_t > 0.45$, we that:

$$\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \geq C - \varepsilon'_t L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)(t-1-t_j) - \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \tag{52}$$

but $\varepsilon'_t = \frac{1}{4tL^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)(t-1-t_j)}$ so:

$$\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \geq C - \frac{1}{4t} - \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \tag{53}$$

But if $\delta_t \leq 0.45$:

$$\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \leq C + \varepsilon'_t L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)(t-1-t_j) + \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \tag{54}$$

or using the definition of ε'_t:

$$\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \leq C + \frac{1}{4t} + \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \tag{55}$$

E.1.2 Masking Procedure:

In order to prevent any leakage of information when the user decrypts the result of this approximate comparison, we use a masking procedure where the algorithm adds a big noise to the bit encrypting the approximation of the comparison, somehow masking its value to the user. In order for this procedure to be secure, the algorithm needs to sample the noise from a distribution such the resulting distribution of the result observed by the user is independent of the value of δ_t (see Prop. 13). Formally, we add a noise $\xi \sim \Xi$ such that for any $x, x' \in [0,1]$:

$$\mathbb{P}\left(Dec_{\mathbf{sk}}(\xi + x)\right) = \mathbb{P}\left(Dec_{\mathbf{pk}}(\xi + x')\right) \tag{56}$$

Finding such distribution is highly dependent on the encryption scheme used and its parameters. In our implementation, we used the CKKS scheme with depth $D = 100$, level of security $\kappa = 128$ and a log size of modulus $\log_2(q_0) = 4982$. Therefore, for a cyphertext ct at a given level l encrypting a number $x \in [0,1]$ with a pair of public and secret key $(\mathbf{pk}, \mathbf{sk})$ we have that:

$$Dec_{\mathbf{sk}}(ct) = \langle ct, \mathbf{sk}\rangle (\mathrm{mod} q_l) \tag{57}$$

with $q_l = 2^l q_0$. When sampling an integer r uniformly in $\{0, \dots, q_l - 1\}$, we have that for any $k \in \{0, \dots, q_l - 1\}$ the distribution of $r + k (\text{mod} q_l)$ is uniform over $\{0, \dots, q_l - 1\}$. We leverage this result to creates a masking procedure detailed in Alg. 10

Algorithm 10 Masking Procedure

```
**Input:** ciphertext: $ct$, modulus factor: $q_l$, cyclotomial polynomial degree: $M$
Sample uniformly $r$ in $\{0, \dots, q_l - 1\}$
Compute the polynomial $\widetilde{r} \in \mathbb{Z}[X]/(X^M + 1)$ such that $\widetilde{r}(X) = r$
**Return:** $ct + \widetilde{r}$
```

Upon receiving the decryption of $ct + \widetilde{r}$, the unmasking procedure is consists in simply subtracting r.

E.1.3 Impact on the Growth of the determinant:

In this section, we study the impact on the determinant of the design matrix when Condition 8 is satisfied for some constant C'. Our result is based on the classic following lemma.

Lemma 14. *For any positive definite symmetric matrix A, B and symmetric semi-positive definite matrix C such that $A = B + C$ we have that:*

$$\frac{det(A)}{det(B)} \geq 1 + Tr\left(B^{-1/2} C B^{-1/2}\right) \tag{58}$$

Proof. of Lemma 14. Using that $A = B + C$:

$$\frac{\det(A)}{\det(B)} = \det\left(I_d + B^{-1/2} C B^{-1/2}\right) \geq 1 + \operatorname{Tr}\left(B^{-1/2} C B^{-1/2}\right) = 1 + \operatorname{Tr}\left(B^{-1} C\right) \tag{59}$$

The last inequality is a consequence of the following inequality:

$$\forall n \in \mathbb{N}^\star, \forall a \in \mathbb{R}_+^n, \qquad 1 + \sum_{i=1}^n a_i \leq \prod_{i=1}^n (1 + a_i) \tag{60}$$

Indeed $I_d + B^{-1/2} C B^{-1/2}$ is symmetric definite positive hence its eignevalues are positive. □

Therefore, using the lemma above applied to the design matrix, $V_t = \bar{V}_j + \sum_{l=t_j+1}^{t-1} s_{l,a_l} s_{l,a_l}^\intercal$ for $t \geq t_j + 1$, we have that:

$$\det\left(V_t\right) \geq \left(1 + \operatorname{Tr}\left(\bar{V}_j^{-1} \sum_{l=t_j+1}^{t-1} s_{l,a_l} s_{l,a_l}^\intercal\right)\right) \det(\bar{V}_j) \tag{61}$$

E.1.4 Putting Everything Together:

We are finally, ready to prove an upper-bound on the number of batches. First, let's recall Prop. 7.

Proposition. *If $C - \frac{L\eta}{\sqrt{\lambda + L^2}} > \frac{1}{4}$, the number of episodes in Alg. 4, M_T for T steps, is bounded by:*

$$M_T \leq 1 + \frac{d \ln\left(1 + \frac{L^2 T}{\lambda d}\right)}{2 \ln\left(\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}}\right)} + \frac{\ln(T)}{\ln(1 + \eta)} \tag{62}$$

Proof. of Lem. 7. Let's define for $i \geq 1$, the macro-episode:

$$n_i = \min\left\{t > n_{i-1} \mid \delta_t > \varepsilon_t\right\} \tag{63}$$

with $n_0 = 0$. In other words, macro-episodes are episodes such that the norm of the context has grown too big. It means that for all episodes between two macro-episodes the batches are ended because the current batch is too long. Therefore for macro-episode i, thanks to Eq. (51) and Prop. 13:

$$C - \varepsilon'_{n_i} L^2 \left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)(n_i - 1 - t_j) - \frac{L(n_i - 1 - t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \leq \operatorname{Tr}\left(\bar{V}_j^{-1} \sum_{l=t_j+1}^{n_i-1} s_{l,a_l} s_{l,a_l}^\intercal\right) \tag{64}$$

where $\varepsilon'_{n_i} = \left(4 n_i L^2 \left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)(n_i - 1 - t_j)\right)^{-1}$ as defined in Alg. 4. But the batch j for which $n_i = t_{j+1}$ is such that $t_{j+1} - t_j \leq \eta t_j + 1$ (thanks to the second if condition in Alg. 4). Hence:

$$\frac{L(n_i - 1 - t_j)}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \leq \frac{L\eta}{\sqrt{t_j(\lambda + L^2 t_j)}} \leq \frac{L\eta}{\sqrt{\lambda + L^2}} \tag{65}$$

Therefore by Lem. 14 we have that:

$$\det \bar{V}_{j+1} \geq \left(1 + C - \left(\frac{1}{4} + \frac{L\eta}{\sqrt{\lambda + L^2}}\right)\right)\det \bar{V}_j \tag{66}$$

because for all t, $\varepsilon'_t L^2 \left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)(t - 1 - t_j) \leq \frac{1}{4}$ and because for any episode j between two macro-episodes i and $i+1$ the determinant of the design matrix is an increasing function of the episode (because for two matrices M, N symmetric semi-definite positive $\det(M + N) \geq \det(M)$).

Thus $\det \bar{V}_{j_i} \geq (\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}})\det \bar{V}_{j_{i-1}}$ where j_i is the episode such that $n_i = t_{j_i+1}$. Therefore the number of macro-episodes M_1 is such that:

$$\left(\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}}\right)^{M_1 - 1} \leq \frac{\det(\bar{V}_{M_T})}{\det(\bar{V}_0)} \tag{67}$$

where $\bar{V}_{M_T}$ is the design matrix after T steps (or M_T batches) and $\bar{V}_0 = \lambda I_d$. This upper bound gives that:

$$M_1 \leq 1 + \frac{\ln\left(\frac{\det(\bar{V}_{M_T})}{\det(\bar{V}_0)}\right)}{\ln\left(\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}}\right)} \tag{68}$$

if $\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}} > 1$. Moreover, thanks to Lemma 10 in Abbasi-Yadkori et al. (2011), the log-determinant of the design matrix is bounded by: $\ln\left(\frac{\det(\bar{V}_{M_T})}{\det(\bar{V}_0)}\right) \leq d \ln\left(1 + \frac{TL^2}{\lambda d}\right)$. In addition, there is at most $1 + \frac{\ln(n_{i+1}/n_i)}{\ln(1+\eta)}$ batches between macro-episode i and $i+1$. Therefore:

$$\begin{aligned} M_T &\leq \sum_{i=0}^{M_1 - 1} 1 + \frac{1}{\ln(1+\eta)}\ln(n_{i+1}/n_i) = M_1 + \frac{\ln(T)}{\ln(1+\eta)} \\ &\leq 1 + \frac{d \ln\left(1 + \frac{L^2 T}{\lambda d}\right)}{2\ln\left(\frac{3}{4} + C - \frac{L\eta}{\sqrt{\lambda + L^2}}\right)} + \frac{\ln(T)}{\ln(1+\eta)} \end{aligned}$$

□

E.2 Regret Upper Bound (Proof of Thm. 8)

Now that we have shown an upper-bound on the number of bathes for the HELBA algorithm, we are ready to prove the regret bound of Thm. 8. The proof of this theorem follows the same logic as the regret analysis of OFUL. That is to say, we first show a high-probability upper bound on the regret thanks to optimism and then proceed to bound each term of the bonus used in HELBA.

We first show the following lemma giving a first upper bound on the regret relating the error due to the approximation of the argmax and optimism.

Lemma 15. *For any $\delta > 0$, the regret of Alg. 4 is bounded with probability at least $1-\delta$ by:*

$$\begin{aligned} R_T(\text{HELBA}) \leq & \underbrace{\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \frac{4}{t}\left(1+2\widetilde{\beta}(j)\left[\frac{2}{t}+\sqrt{\frac{L}{t_j^{3/2}\sqrt{t_jL^2+\lambda}}}+L\sqrt{\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}}\right]\right)}_{:=①} \\ & + \underbrace{\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} 2\widetilde{\beta}(j)\left[\text{sqrt}_{\text{HE}}\left(s_{t,a}^\top Dec_{\mathbf{sk}}(\bar{A}_j)s_{t,a}+\frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}\right)+\frac{1}{t}\right]}_{:=②} \end{aligned} \tag{69}$$

where for every time step t, $a_t^\star = \arg\max_{a\in[K]}\langle x_{t,a}, \theta^\star\rangle$, M_T is the number of batches and $R_T(\text{HELBA}) = \sum_{t=1}^T \langle \theta^\star, s_{t,a_t^\star} - s_{t,a_t}\rangle$.

Proof. of Lem. 15. First, let's define E the event that all confidence ellipsoids, $\tilde{\mathcal{C}}_j$, contain $\theta^\star$ with probability at least $1-\delta$. That is to say $E = \left\{\theta^\star \in \bigcap_{j=1}^{+\infty}\tilde{\mathcal{C}}_j(\delta)\right\}$. Thanks to Prop. 9, $\mathbb{P}(E) \geq 1-\delta$. Because E is included in the event described by Prop. 9.

Therefore conditioned on the event E, after T steps the regret can be decomposed as:

$$\begin{aligned} R_T(\text{HELBA}) = \sum_{t=1}^T & \langle \theta^\star, s_{t,a_t^\star}\rangle - \max_{a\leq K}\text{Dec}_{\mathbf{sk}}(\rho_a(t)) + \max_{a\leq K}\text{Dec}_{\mathbf{sk}}(\rho_a(t)) - \text{Dec}_{\mathbf{sk}}(\rho_{a_t}(t)) \\ & + \text{Dec}_{\mathbf{sk}}(\rho_{a_t}(t)) - \langle\theta^\star, s_{t,a_t}\rangle \end{aligned} \tag{70}$$

where $\rho_a(t)$ is the optimistic upper bound on the reward of arm a computed by Alg. 4 and $a_t^\star = \arg\max_{a\in[K]}\langle\theta^\star, s_{t,a_t^\star}\rangle$. Now for any $t \leq T$, under the event E, $\langle\theta^\star, s_{t,a_t^\star}\rangle \leq \max_{a\leq K}\text{Dec}_{\mathbf{sk}}(\rho_a(t))$. But thanks to Cor. 5, for any $t\geq 1$ inside batch j:

$$\max_a \text{Dec}_{\mathbf{sk}}(\rho_a(t)) - \text{Dec}_{\mathbf{sk}}(\rho_{a_t}(t)) \leq \frac{1}{t}\left(1+\widetilde{\beta}(j)\left[\frac{2}{t}+\sqrt{\frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}}+L\sqrt{\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}}\right]\right) \tag{71}$$

In addition, we have that under event E:

$$\text{Dec}_{\mathbf{sk}}(\rho_{a_t}(t)) - \langle\theta^\star, s_{t,a_t}\rangle = \langle\tilde{\theta}_j - \theta^\star, s_{t,a_t}\rangle + \widetilde{\beta}(j)\left[\text{sqrt}_{\text{HE}}\left(s_{t,a}^\top\text{Dec}_{\mathbf{sk}}(\bar{A}_j)s_{t,a}+\frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}\right)+\frac{1}{t}\right]$$

But still conditioned on the event E, $\langle\tilde{\theta}_j - \theta^\star, s_{t,a_t}\rangle \leq \|\tilde{\theta}_j - \theta^\star\|_{\bar{V}_j}\|s_{t,a_t}\|_{\bar{V}_j^{-1}} \leq \widetilde{\beta}(j)\|s_{t,a_t}\|_{\bar{V}_j^{-1}} \leq \widetilde{\beta}(j)\left[\text{sqrt}_{\text{HE}}\left(s_{t,a_t}^\top\text{Dec}_{\mathbf{sk}}(\bar{A}_j)s_{t,a_t}+\frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}\right)+\frac{1}{t}\right]$. Putting the last two equations together, for every step $t\leq T$:

$$\begin{aligned} \langle\theta^\star, s_{t,a_t^\star}-s_{t,a_t}\rangle \leq & \frac{1}{t}\left(1+\widetilde{\beta}(j)\left[\frac{2}{t}+\sqrt{\frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}}+L\sqrt{\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}}\right]\right) \\ & + 2\widetilde{\beta}(j)\left[\text{sqrt}_{\text{HE}}\left(s_{t,a_t}^\top\text{Dec}_{\mathbf{sk}}(\bar{A}_j)s_{t,a_t}+\frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}\right)+\frac{1}{t}\right] \end{aligned}$$

□

Bounding ①. We now proceed to bound each term in Eq. (69). The following lemma is used to ①.

Lemma 16. *For all $t\geq 1$:*

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}}\frac{1}{t}\left(1+\widetilde{\beta}(j)\left[\frac{2}{t}+\sqrt{\frac{L}{t_j^{3/2}\sqrt{t_jL^2+\lambda}}}+L\sqrt{\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}}\right]\right) \leq \mathcal{O}(\ln(T)^{3/2}) \tag{72}$$

Proof. of Lem. 16. Because $t_j \geq 1$:

$$\frac{L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} \leq \frac{L}{\sqrt{\lambda}}, \quad \widetilde{\beta}(j) \leq 1 + \sqrt{\lambda} S + \sigma\sqrt{d\left(\ln\left(1 + \frac{L^2 T}{\lambda d}\right) + \ln\left(\frac{\pi^2 T^2}{6\delta}\right)\right)} \tag{73}$$

Bounding each component of the sum of ① in Eq. (69) individually, we get:

$$\sum_{j=0}^{M_T - 1}\sum_{t=t_j+1}^{t_{j+1}} \frac{1}{t} \leq (1 + \ln(T)) \tag{74}$$

Hence:

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \frac{1}{t}\left(1 + \widetilde{\beta}(j)\left[\frac{2}{t} + \sqrt{\frac{L}{t_j^{3/2}\sqrt{t_j L^2 + \lambda}}} + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}}\right]\right) \leq (1 + \ln(T))\Bigg[1 + \left(1 + \sqrt{\lambda}S + \sigma\sqrt{d\left(\ln\left(1 + \frac{L^2 T}{\lambda d}\right) + \ln\left(\frac{\pi^2 T^2}{6\delta}\right)\right)}\right)\left(2 + L\sqrt{\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}} + \sqrt{\frac{L}{\sqrt{\lambda}}}\right)\Bigg]$$

□

Lem. 16 shows that the error from our procedure to select the argmax induces only an additional logarithmic cost in T compared with the regret of directly selecting the argmax of the UCBs $(\rho_a(t))_{a \leq K}$.

Bounding ②. We are now left with bounding the second term in Eq. (69). This term is usually the one that appears in regret analysis for linear contextual bandits. First, ② can be further broke down thanks to the following lemma.

Lemma 17. *For all $t \geq 1$,*

$$\begin{aligned}\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \widetilde{\beta}(j)\left[\mathrm{sqrt}_{\mathrm{HE}}\left(s_{t,a_t}^\top Dec_{\mathbf{sk}}(\bar{A}_j) s_{t,a_t} + \frac{L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}}\right) + \frac{1}{t}\right] &\leq \underbrace{\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \frac{2\widetilde{\beta}(j)}{t}}_{:=ⓐ} \\ &+ \underbrace{\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \widetilde{\beta}(j)\|s_{t,a_t}\|_{\bar{V}_j^{-1}}}_{:=ⓑ} + \underbrace{\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \widetilde{\beta}(j)\sqrt{\frac{2L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}}}}_{:=ⓒ}\end{aligned} \tag{75}$$

Proof. of Lem. 17. For any time $t \geq 1$ thanks to Prop. 4, we have:

$$\begin{aligned}\mathrm{sqrt}_{\mathrm{HE}}\left(s_{t,a_t}^\top \mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j) s_{t,a_t} + \frac{L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}}\right) &\leq \frac{1}{t} + \sqrt{\|s_{t,a_t}\|^2_{\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)} + \frac{L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}}} \\ &\leq \frac{1}{t} + \sqrt{\frac{L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} + \|s_{t,a_t}\|^2_{\bar{V}_j^{-1}} + \|s_{t,a_t}\|_2^2\|\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j) - \bar{V}_j^{-1}\|} \\ &\leq \frac{1}{t} + \sqrt{\frac{2L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}} + \|s_{t,a_t}\|^2_{\bar{V}_j^{-1}}} \\ &\leq \frac{1}{t} + \sqrt{\frac{2L}{t_j^{3/2}\sqrt{\lambda + L^2 t_j}}} + \|s_{t,a_t}\|_{\bar{V}_j^{-1}}\end{aligned}$$

□

We proceed to bound each term ⓐ, ⓑ, ⓒ. Bounding ⓑ is similar to the analysis of OFUL. On the other hand, bounding neatly ⓒ is the reason why we introduced the condition that a new episode is started is $t \geq (1+\eta)t_j$.

The following lemma bounds ⓐ which is simply a numerical error due to the approximation of the square root.

Lemma 18. *For any $T \geq 1$,*

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \frac{4\widetilde{\beta}(j)}{t} \leq 4\left(1+\sqrt{\lambda}S+\sigma\sqrt{d\left(\ln\left(1+\frac{L^2T}{\lambda d}\right)+\ln\left(\frac{\pi^2T^2}{6\delta}\right)\right)}\right)(1+\ln(T)) \tag{76}$$

Proof. of Lem. 18. Using the upper bound on the $\widetilde{\beta}(j)$ shown in the proof of Lem. 16, we get the result. □

We are finally left with the two terms ⓑ and ⓒ. The first term, ⓑ, will be compared to the bonus used in OFUL so that we can use Lemma 11 in Abbasi-Yadkori et al. (2011) to bound it. But first, we need to show how the norm for two different matrices A and B relates to each other.

Lemma 19. *For any context $x \in \mathbb{R}^d$ and symmetric semi-definite matrix A,B and C such that $A = B + C$ then:*

$$\|x\|^2_{B^{-1}} \leq \lambda_{\max}\left(I_d + B^{-1/2}CB^{-1/2}\right)\|x\|^2_{A^{-1}} \leq \left(1 + Tr\left(B^{-1/2}CB^{-1/2}\right)\right)\|x\|^2_{A^{-1}} \tag{77}$$

where $\lambda_{\max}(.)$ returns the maximum eigenvalue of a matrix.

Proof. of Lemma 19. We have by definition of A and B:

$$\begin{aligned}\langle x, A^{-1}x\rangle = \langle x, (B+C)^{-1}x\rangle &= \langle x, B^{-1/2}(I_d + B^{-1/2}CB^{-1/2})^{-1}B^{-1/2}x\rangle & (78)\\ &= \langle B^{-1/2}x, (I_d + B^{-1/2}CB^{-1/2})^{-1}(B^{-1/2}x)\rangle & (79)\\ &\geq \lambda_{\min}\left((I_d + B^{-1/2}CB^{-1/2})^{-1}\right)\|B^{-1/2}x\|^2 & (80)\\ &\geq \frac{1}{\lambda_{\max}(I_d + B^{-1/2}CB^{-1/2})}\|x\|^2_{B^{-1}} & (81)\end{aligned}$$

Hence:

$$\|x\|^2_{B^{-1}} \leq \lambda_{\max}\left(I_d + B^{-1/2}CB^{-1/2}\right)\|x\|^2_{A^{-1}} \tag{82}$$

The result follows from Weyl's inequality Horn and Johnson (1991), that is to say for all symmetric matrix M, N $\lambda_{\max}(M+N) \leq \lambda_{\max}(M) + \lambda_{\max}(N)$. And the fact that all eigenvalues of $B^{-1/2}CB^{-1/2}$ are positive hence $\lambda_{\max}(B^{-1/2}CB^{-1/2}) \leq \mathrm{Tr}(B^{-1/2}CB^{-1/2}) = \mathrm{Tr}(CB^{-1})$. □

We are now able to bound ⓑ using Lemma 11 in Abbasi-Yadkori et al. (2011).

Lemma 20. *If $\lambda \geq L^2$ we have:*

$$\begin{aligned}\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} 2\beta(j)\|s_{t,a_t}\|_{\bar{V}_j^{-1}} \leq \beta^\star\sqrt{2d\ln\left(1+\frac{TL^2}{\lambda d}\right)}\Bigg[&\sqrt{T\left(1.25+C+L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)\right)}\\ &+\sqrt{M_T\left(\eta^2+\frac{L}{(\lambda+L^2)^{3/2}}\right)}\Bigg]\end{aligned} \tag{83}$$

with $\beta^\star = 1+\sqrt{\lambda}S+\sigma\sqrt{d\left(\ln\left(1+\frac{L^2T}{\lambda d}\right)+\ln\left(\frac{\pi^2T^2}{6\delta}\right)\right)}$

Proof. of Lem. 20. For any time t in batch j, we have thanks to Lem. 19 that:

$$\|s_{t,a_t}\|_{\bar{V}_j^{-1}} \leq \sqrt{1+\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)}\|s_{t,a_t}\|_{V_t^{-1}} \tag{84}$$

with $V_t = \lambda I_d + \sum_{l=1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal$. The rest of the proof relies on bounding $\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right)$. To do so, we will use the following inequality, see Eq. (51):

$$-\frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}} \leq \mathrm{Tr}\left((\bar{V}_j^{-1} - \mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j))\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^\intercal\right) \leq \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}$$

Therefore$\delta_t \leq 0.45$ during batch j, because it is not over while no condition is satisfied. Thanks to Prop. 13 with $\varepsilon' = \frac{1}{4tL^2(t-1-t_j)}$, we get:

$$\forall t \in \{t_j+1, \dots t_{j+1}-1\}, \qquad \mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C + \frac{t-1-t_j}{4t}$$

However, for $t = t_{j+1}$ we have either that $\delta_{t_{j+1}} > 0.45$ or $t_{j+1} \geq (1+\eta)t_j$:

- If $\delta_{t_{j+1}} \leq 0.45$, then $\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t_{j+1}-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C + \frac{t_{j+1}-1-t_j}{4t}$
- If $\delta_{t_{j+1}} > 0.45$, then $\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t_{j+1}-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \geq C - \frac{t_{j+1}-1-t_j}{4t}$ but $\delta_{t_{j+1}-1} \leq 0.45$ thus $\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t_{j+1}-2} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C + \frac{t-1-t_j}{4t}$. Therefore, using that $\|s_{t,a_t}\|^2_{\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)} \leq \lambda_{\max}(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j))\|s_{t,a_t}\|_2^2 \leq L^2\left(\frac{1}{\lambda} + \frac{1}{L\sqrt{\lambda+L^2}}\right)$. Hence, we have that:

$$\mathrm{Tr}\left(\mathrm{Dec}_{\mathbf{sk}}(\bar{A}_j)\sum_{l=t_j+1}^{t_{j+1}-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C + \frac{t-1-t_j}{4t} + L^2\left(\frac{1}{\lambda} + \frac{1}{L\sqrt{\lambda+L^2}}\right) \tag{85}$$

To sum up, for all $t_j+1 \leq t \leq t_{j+1}$:

$$\mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right) \leq C + \frac{t-1-t_j}{4t} + L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right) + \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}} \tag{86}$$

Overall, we have that:

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \|s_{t,a_t}\|_{\bar{V}_j^{-1}} \leq \sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \sqrt{1 + \mathrm{Tr}\left(\bar{V}_j^{-1}\sum_{l=t_j+1}^{t-1} s_{l,a_l}s_{l,a_l}^{\intercal}\right)}\|s_{t,a_t}\|_{V_t^{-1}} \tag{87}$$

$$\leq \sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \| s_{t,a_t}\|_{V_t^{-1}}\sqrt{1 + C + \frac{t-1-t_j}{4t} + L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right) + \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}} \tag{88}$$

$$\leq \sqrt{\frac{5}{4} + C + L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)}\sqrt{T\sum_{t=1}^{T}\|s_{t,a_t}\|^2_{V_t^{-1}}} + \sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \| s_{t,a_t}\|_{V_t^{-1}}\sqrt{\frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}} \tag{89}$$

where the last inequality is due to Cauchy-Schwarz inequality. The first term in inequality Eq. (89) is bounded by using Lemma 29 in Ruan et al. (2020),

$$\sum_{t=1}^{T}\|x_{t,a_t}\|^2_{V_t^{-1}} \leq 2\ln\left(\frac{\det(V_T)}{\det(V_0)}\right) \leq 2d\ln\left(1 + \frac{TL^2}{\lambda d}\right) \tag{90}$$

In addition, the last term in Eq. (89) is bounded by:

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \frac{L(t-1-t_j)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}} \leq \sum_{j=0}^{M_T-1}\frac{L(t_{j+1}-t_j)^2}{2t_j^{3/2}\sqrt{\lambda+L^2t_j}} \tag{91}$$

But a consequence of the second condition is that the length of batch j satisfies $t_{j+1}-t_j \leq \eta t_j + 1$. Therefore:

$$\sum_{j=0}^{M_T-1}\frac{Lf(t_{j+1}-t_j)^2}{2t_j^{3/2}\sqrt{\lambda+L^2t_j}} \leq \sum_{j=0}^{M_T-1}\frac{L(\eta t_j+1)^2}{2t_j^{3/2}\sqrt{\lambda+L^2t_j}} \tag{92}$$

$$\leq \sum_{j=0}^{M_T-1}\frac{L(\eta^2t_j^2+1)}{t_j^{3/2}\sqrt{\lambda+L^2t_j}} \tag{93}$$

$$\leq \sum_{j=0}^{M_T-1}\eta^2 + \frac{L}{(\lambda+L^2)^{3/2}} \leq \eta^2 M_T + \frac{LM_T}{(\lambda+L^2)^{3/2}} \tag{94}$$

Putting everything together we get:

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} \|s_{t,a_t}\|_{\bar{V}_j^{-1}} \leq \sqrt{\frac{5}{4} + C + L^2\left(\frac{1}{\lambda} + \frac{1}{\sqrt{\lambda}}\right)}\sqrt{2Td\ln\left(1+\frac{TL^2}{\lambda d}\right)} + \sqrt{2d\ln\left(1+\frac{TL^2}{\lambda d}\right)\left(\eta^2 M_T + \frac{LM_T}{(\lambda+L^2)^{3/2}}\right)}$$

Hence the result using the upper bound on $\widetilde{\beta}(j)$ proved in Lem. 16. □

Finally, the last term to bound is ⓒ, that we do similarly to the end of the proof of Lem.20.

Lemma 21. *For all $T \geq 1$,*

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} 2\widetilde{\beta}(j)\sqrt{\frac{2L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}} \leq 2\sqrt{2L}M_T\beta^\star\left[1+\frac{\eta}{\sqrt{L}}\right] \tag{95}$$

with $\beta^\star = 1 + \sqrt{\lambda}S + \sigma\sqrt{d\left(\ln\left(1+\frac{L^2T}{\lambda d}\right) + \ln\left(\frac{\pi^2T^2}{6\delta}\right)\right)}$ and M_T the number of episodes.

Proof. of Lem. 21. We have:

$$\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} 2\widetilde{\beta}(j)\sqrt{\frac{2L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}} \leq 2\sqrt{2L}\max_j \widetilde{\beta}(j)\sum_{j=0}^{M_T-1}\sqrt{\frac{1}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}}(t_{j+1}-t_j) \tag{96}$$

But the condition on the length of the batch ensures that for any batch j, $t_{j+1} - t_j \leq \eta t_j + 1$, thus equation above can be bounded by:

$$\begin{aligned}\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} 2\widetilde{\beta}(j)\sqrt{\frac{2L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}} &\leq 2\sqrt{2L}\max_j \widetilde{\beta}(j)\left[M_T + \sum_{j=0}^{M_T-1}\eta\sqrt{\frac{\sqrt{t_j}}{\sqrt{\lambda+L^2t_j}}}\right] &(97)\\ &\leq 2\sqrt{2L}\max_j \widetilde{\beta}(j)\left[M_T + \sum_{j=0}^{M_T-1}\frac{\eta}{\sqrt{L}}\right] &(98)\\ &\leq 2\sqrt{2L}M_T\max_j \widetilde{\beta}(j)\left[1+\frac{\eta}{\sqrt{L}}\right] &(99)\end{aligned}$$

Hence the result. □

Finally, we can finish the proof of Thm. 8, but we first recall its statement.

Theorem. *Under Asm. 1, for any $\delta > 0$ and $T \geq d$, there exists universal constants $C_1, C_2 > 0$ such that the regret of* HELBA *(Alg. 4) is bounded with probability at least $1-\delta$ by:*

$$R_T \leq C_1\beta^\star\left(\sqrt{\left(\frac{5}{4}+C\right)dT\ln\left(\frac{TL}{\lambda d}\right)} + \frac{L^{3/2}}{\sqrt{\lambda}}\ln(T)\right) + C_2\beta^\star M_T\max\left\{\sqrt{L}+\eta, \eta^2 + \frac{L}{\sqrt{\lambda+L^2}^3}\right\}$$

Proof. of Thm. 8. For any $\delta > 0$, let's define the event E as in the proof of Lem. 15. Then conditioned on this event, we have using Lem. 15:

$$\begin{aligned}R_T(\text{HELBA}) \leq &\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}}\frac{4}{t}\left(1+2\widetilde{\beta}(j)\left[\frac{2}{t}+\sqrt{\frac{L}{t_j^{3/2}\sqrt{t_jL^2+\lambda}}} + L\sqrt{\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}}\right]\right)\\ &+\sum_{j=0}^{M_T-1}\sum_{t=t_j+1}^{t_{j+1}} 2\widetilde{\beta}(j)\left[\text{sqrt}_{\text{HE}}\left(s_{t,a_t}^\top \text{Dec}_{\mathbf{sk}}(\bar{A}_j)s_{t,a_t} + \frac{L}{t_j^{3/2}\sqrt{\lambda+L^2t_j}}\right)+\frac{1}{t}\right]\end{aligned} \tag{100}$$

But using Lem. 16 to bound the first term of the RHS equation above but also Lem. 17, 18, 20 and 21 to the bound the second term, we get:

$$R_T\,(\text{HELBA}) \leq (1+\ \ln(T))\left[1+\beta^\star\left(6+L\sqrt{\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}}+\sqrt{\frac{L}{\sqrt{\lambda}}}\right)\right]$$
$$+\beta^\star\sqrt{2d\ln\left(1+\frac{TL^2}{\lambda d}\right)}\left[\sqrt{\frac{5}{4}+C+L^2\left(\frac{1}{\lambda}+\frac{1}{\sqrt{\lambda}}\right)}\sqrt{T}+\left(\eta^2+\frac{L}{(\lambda+L^2)^{3/2}}\right)M_T\right] \tag{101}$$
$$+2\sqrt{2L}M_T\beta^\star\left[1+\frac{\eta}{\sqrt{L}}\right]$$

with $\beta^\star = 1+\sqrt{\lambda}S+\sigma\sqrt{d\left(\ln\left(1+\frac{L^2T}{\lambda d}\right)+\ln\left(\frac{\pi^2T^2}{6\delta}\right)\right)}$ and $M_T = 1+\frac{d\ln\left(1+\frac{L^2T}{\lambda d}\right)}{2\ln\left(\frac{3}{4}+C-\frac{L\eta}{\sqrt{\lambda+L^2}}\right)}+\frac{\ln(T)}{\ln(1+\eta)}$ □

F IMPLEMENTATION DETAILS:

In this section, we further detail how HELBA is implemented. In particular, we present the matrix multiplication and matrix-vector operations.

For the experiments, we used the PALISADE library (development version v1.10.4) PAL (2020). This library automatically chooses most of the parameters used for the CKKS scheme. In particular the ring dimension of the ciphertext space is chosen automatically. In the end, the user only need to choose four parameters: the maximum multiplicative depth (here chosen at 100), the number of bits used for the scaling factor (here 50), the batch size that is to say the number of plaintext slots used in the ciphertext (here 8) and the security level (here chosen at 128 bits for Fig. 1).

F.1 Matrix/Vector Encoding

Usually, when dealing with matrices and vectors in homomorphic encryption there are multiple ways to encrypt those. For example, with a vector $y \in \mathbb{R}^d$ one can create d ciphertexts encrypting each value y_i for all $i \leq d$. This approach is nonetheless expensive in terms of memory. An other approach is to encrypt directly the whole vector in a single ciphertext. A ciphertext is a polynomial ($X \mapsto \sum_{i=0}^N a_i X^i$) where each coefficient is used to encrypt a value of y ($a_i = y_i$ for $i \leq d$). This second method is oftentimes preferred as it reduce memory usage.

It is possible to take advantage of this encoding method in order to facilitate computations, e.g., matrix multiplication, matrix-vector operation or scalar product. In this work, we need to compute the product of square matrices of size $d \times d$, thus we choose to encrypt each matrix/vector as a unique ciphertext (assuming $d \leq N$). We have two different encoding for matrices and vectors. For a matrix $A = (a_{i,j})_{i\in\{0,\dots,p-1\},j\in\{0,\dots,q-1\}}$ with $p, q \in \mathbb{N}$, we first transform A into a vector of size pq, $\tilde{a} = (a_{0,0}, a_{0,1}, \dots, a_{0,q-1}, \dots, a_{1,0}, \dots, a_{1,q-1}, \dots, a_{p-1,q-1})$. This vector is then encrypted into a single ciphertext. But for a vector $y \in \mathbb{R}^q$, we create a bigger vector of dimension pq (here p is a parameter of the encoding method for vectors), $\tilde{y} = (y_j)_{i\in\{0,\dots,p-1\},j\in\{0,\dots,q-1\}} = (y_0, \dots, y_{q-1}, y_0, \dots, y_{q-1}, \dots, y_{q-1})$. We choose those two encodings because the homomorphic multiplication operation of PALISADE only perform a coordinate-wise multiplication between two ciphertexts. Therefore, using this encoding, a matrix-vector product for a matrix $A \in \mathbb{R}^{p\times q}$, a vector $y \in \mathbb{R}^q$, a public key **pk** can be computed as:

$$c_A \times c_y = \text{Enc}_{\mathbf{pk}}(\tilde{a}\cdot\tilde{y}) = \text{Enc}_{\mathbf{pk}}\Big(\Big(a_{0,0}y_0, a_{0,1}y_1, \dots, a_{0,q-1}y_{q-1}, a_{1,0}y_0, a_{1,1}y_1, \dots, a_{1,q-1}y_{q-1},$$
$$\dots, a_{p-1,q-1}y_{q-1}\Big)\Big)$$

with $\tilde{a}$ the encoding of A, $c_A = \text{Enc}_{\mathbf{pk}}(\tilde{a})$, $\tilde{y}$ the encoding of y of dimension pq and $c_y = \text{Enc}_{\mathbf{pk}}(\tilde{y})$, $\times$ the homomorphic multiplication operation and $\tilde{a}\cdot\tilde{y}$ the element-wise product. Then using EvalSumCol (an implementation of the SumColVec method from Han et al. in the PALISADE library) to compute partial sums of the coefficients of $c_A \times c_y$, we get:

$$\text{EvalSumCol}(c_A \times c_y, p, q) = \text{Enc}_{\mathbf{pk}}\left(\left(\sum_{j=0}^{q-1} a_{0,j}y_j, \dots, \sum_{j=0}^{q-1} a_{0,j}y_j, \sum_{j=0}^{q-1} a_{1,j}y_j, \dots, \sum_{j=0}^{q-1} a_{1,j}y_j, \dots, \sum_{j=0}^{q-1} a_{p-1,j}y_j\right)\right)$$

Finally, the matrix-vector product Ay is computed by EvalSumCol$(c_A \times c_y, p, q)$ taking the coefficient at $(j + j \cdot p)_{j \in [p]}$.

F.2 Matrix Multiplication

Using the encoding of App. F.1 we have a way to compute a matrix-vector product therefore computing the product between two square matrices $M, N \in \mathbb{R}^{p \times p}$ can be done using a series of matrix-vector products. However, this approach requires p ciphertexts to represent a matrix. We then prefer to use the method introduced in Sec. 3 of Jiang et al. (2018). This method relies on the following identity for any matrices $M, N \in \mathbb{R}^{p \times p}$ and $i, j \in \{0, \dots, p-1\}$:

$$\begin{aligned}(MN)_{i,j} &= \sum_{k=0}^{p-1} M_{i,k}N_{k,j} \\ &= \sum_{k=0}^{p-1} M_{i,[i+k+j]_p}N_{[i+k+j]_p,j} \\ &= \sum_{k=0}^{p-1} \sigma(M)_{i,[j+k]_p}\tau(N)_{[i+k]_p,j} \\ &= \sum_{k=0}^{p-1} (\phi^k \circ \sigma(M))_{i,j}(\psi^k \circ \tau(N))_{i,j}\end{aligned} \tag{102}$$

where we define σ, τ, ψ and ϕ as: and $[.]_p$ is the modulo operator. Therefore, using Eq. (102) we have that

- $\sigma(M)_{i,j} = M_{i,[i+j]_p}$
- $\tau(M)_{i,j} = M_{[i+j]_p,j}$
- $\psi(M)_{i,j} = M_{i,[i+1]_p}$
- $\phi(M)_{i,j} = M_{[i+1]_p,j}$

computing the product between M and N can simply be done by computing a component-wise multiplication between $(\phi^k \circ \sigma(M))_{i,j}$ and $(\psi^k \circ \tau(N))_{i,j}$ for all $k \in \{0, \dots, p-1\}$. Those quantities can in turn be easily computed thanks to a multiplication between a plaintext and a ciphertext (this does not impact the depth of the ciphertext).

F.3 Influence of the Security Level

Finally, we investigate the influence of the security level κ on the running time and regret of HELBA. As mentioned in Sec. 6 the security parameter κ ensures that an attacker has to perform at least 2^κ operations in order to decrypt a ciphertext encrypted using an homomorphic encryption scheme. But, the security parameter also has an impact on the computational efficiency of our algorithm. Indeed the dimension N of the ciphertext space, i.e., the degree of the polynomials in $\mathbb{Z}[X]/(X^N + 1)$, increases with the multiplicative depth D and κ. However, this means that our algorithm has to compute operations with polynomials of higher dimensions hence more computationally demanding.

The library PALISADE allows us to choose $\kappa \in \{128, 192, 256\}$. We executed HELBA with the same parameter and the same environment of Sec. 6 except for the parameter κ which now varies in $\{128, 192, 256\}$. First, we investigate the regret of for each parameter κ, this parameter should have no impact on the regret HELBA, as showed in Fig. 2.

Second, we investigate the running time for each $\kappa \in \{128, 192, 256\}$. Table 1 shows the ratio between the total computation time of 130 steps using the environment described in Sec. 6 with HELBA for different security

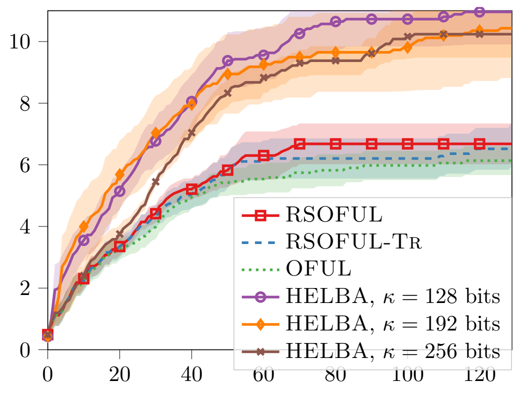

Figure 2: Regret of HELBA for $\kappa \in \{128, 192, 256\}$

Table 1: Ratio of running time for HELBA as a function of κ for the bandit problem of Sec. 6. We use the running time with $\kappa = 128$ bits and $T = 130$ steps as a reference to compute the ratio between this time and the total time for $\kappa \in \{192, 256\}$.

κ (Bits)	Ratio Execution Time
128	1
192	1.016
256	1.026

parameters and $\kappa = 128$ bits. In order to investigate only the effect of the parameter κ, the results in Table 1 are expressed as a ratio. For reference, the total time for $T = 130$ steps and $\kappa = 128$ bits was 20 hours and 39 minutes. As we observe in Table 1 the impact on the security parameter is around 1% and 2% of the total computation time for 128 bits. This increase in computation time represents between 20 and 40 minutes of computation which in some applications can be prohibitive.