Journal of Systems Architecture 182 (2027) 104040

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Fully homomorphic encryption on LlaMA 3 model for partial privacy preserving LLM inference☆

Anes Abdennebi[1,*], Nadjia Kara[2], Lahlou Laaziz[3]

Software and IT Engineering Department, École de Technologie Supérieure, Montreal, H3C1K3, Quebec, Canada

ARTICLE INFO

Keywords:
FHE
Cybersecurity
Post-quantum cryptography
Generative AI
Large language models

ABSTRACT

Generative Artificial Intelligence applications have delivered significant improvements in service efficiency and language understanding. However, the surge in Large Language Model deployment raises critical security and privacy concerns regarding data exposure and unauthorized access to sensitive information. Quantum computing attacks pose an imminent threat to existing cryptographic protections, necessitating post-quantum cryptographic approaches. We present a partially privacy-preserving inference pipeline for LLaMA-3 by integrating fully homomorphic encryption into the transformer's Multi-Query Attention mechanism. Our approach encrypts attention weights and computations across all 32 decoder layers of LLaMA-3.2-3B using lattice-based cryptography via the concrete-ml library, while feed-forward sublayers, layer normalization, and output projections remain in plaintext, a deliberate trade-off achieving practical feasibility on commodity CPUs.

Evaluation on the instruction-tuned LLaMA-3.2-3B model yields: (i) strong semantic preservation (BERTScore-F1 ∼84.4%), (ii) controlled perplexity degradation (1.18 FP16 vs. 2.09 FHE), (iii) competitive BLEU-4 and ROUGE-L scores (∼35.6% and ∼51.8%), (iv) downstream GLUE benchmark accuracy, and (v) 825 ms latency at 17 tokens/second on AMD EPYC CPUs.

We further present quantization-aware FHE and quantization bit-width ablation studies to disentangle quantization effects from FHE noise, demonstrating that partial FHE-secured LLM inference is practically feasible for privacy-sensitive applications while mitigating attacks on attention layers under RLWE hardness assumptions.

1. Introduction

Large Language Models and their building block, the transformer architecture, are the current cutting-edge tools that boosted different Natural Language Processing (NLP) tasks, such as text generation, translation, summarization, sequence classification, and other evolving textual tasks. The transformers, coming with different variants (encoder–decoder, encoder only, or decoder only), depend on the self-attention mechanism implemented in layers forming the majority of the architecture. This mechanism focuses on generating the next words (or tokens) while assigning weights and degrees of relevance to contextual words fed to the model. It is the core engine of any LLM that ensures accurate and pertinent text that makes sense within the scope of the desired task. As LLMs offer their linguistic utility in different tasks, their applicability fits perfectly with the needs of several fields, in which a surge in LLM usage is witnessed in many markets. Large Language Models are now deployed across diverse sectors; market trading, finance, healthcare, and transportation, with these applications processing sensitive data including patient records, financial transactions, and personal identifiers. As LLMs grow prevalent, the risks of data exposure through unencrypted pipelines escalate: processing sensitive information in plaintext introduces substantial privacy and security vulnerabilities.

Attackers and malicious actors seeking the exploitation of such apps (that use LLMs in their backend) tend to use sophisticated techniques to successfully deceive the models into generating the desired outputs in scenarios where their main goal is to extract, distort, or recover sensitive information. The leveraged techniques include prompt injection, jailbreaks, model and data poisoning, data extraction, and supply chain attacks through unauthorized access to the system or API [1–4].

☆ This article is part of a Special issue entitled: 'LLMEI' published in Journal of Systems Architecture.

* Corresponding author.
E-mail addresses: anes.abdennebi.1@ens.etsmtl.ca (A. Abdennebi), Nadjia.Kara@etsmtl.ca (N. Kara), laaziz.lahlou@etsmtl.ca (L. Laaziz).
[1] AI and Security Engineer.
[2] Supervisor.
[3] Co-ordinator.

https://doi.org/10.1016/j.sysarc.2026.104040
Received 9 June 2026; Received in revised form 1 September 2026; Accepted 10 October 2026
Available online 3 November 2026
1383-7621/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Attackers may use one, a few, or a combination of these techniques to achieve their goal [5].

The attacking techniques are not limited to plain queries; they also comprise different vulnerabilities within LLMs. There are multiple ways of manipulating the transformer architecture of LLMs in order to generate malicious content or expose important information about the data used in training the pre-trained model, or even divulge the instructions given to the generative model itself [6–9]. All the previously mentioned risks give attackers an advantage in maximizing the exploitation of their target models. Furthermore, another attack category that falls within the data privacy circle is the Private Information Extraction (PIE) attack, where, in such a scenario, malicious actors can apply techniques to dig deep into LLMs to retrieve information related to any users who had previous interactions (query-answer) with the model. Information including names, birth dates, social insurance numbers, credit cards numbers, passwords, companies confidential records (especially when employees use generative AI chatbots to help them speedup their data analysis, reports generation, or any other time-consuming task), and even extract links between users/clients and their affiliations and/or enterprises [10] potentially leading to a high risk of divulging critical and confidential data.

The risk of violating data privacy in LLMs and their variants, especially when they manifest into existing threats posed by cybercriminals, is severe. It requires stringent techniques to tighten the confidentiality of such data when shared with these vulnerable models. The work in [11] offers a comprehensive framework based on an application security practice (OWASP Top 10 for LLMs), suggesting how to secure LLMs against a wide set of attacks, including *prompt injection*, *insecure output*, *model theft*, and *data poisoning*. However, the work omits the overhead added when applying these theoretically proposed mitigation actions. Moreover, there is no real application that is proven to be effective against one of the mentioned LLM attacks. Another paper by Deng et al. [12] shows the possibility of attacking LLM-based chatbots such as GPT-3.5, GPT-4, and Gemini (previously known as Bard). It bypasses their security measures through jailbreaks and time-based analysis to reverse engineer their defensive mechanisms. The curated attacks by authors' framework (MASTERKEY) achieved an over 21% success rate, leading the chatbots service providers and research community to develop new and enhance existing security strategies and mechanisms based on the outcomes of this work. Many other works are focusing on attacking LLM-based chatbots or tools using jailbreak techniques, experimenting mainly with GPT-3.5, GPT-4, Gemini, Claude, Grok, and other models with natural language processing capabilities, while providing insights on LLMs security reinforcement spots [13–17].

A promising direction to strengthen the model and data's privacy is to use encryption algorithms to encrypt the data at the different levels and stages of the models training and testing, which is the case with many machine learning (ML) and deep learning (DL) models such as RandomForest (RF), Decision Trees (DT), XGBoost (XGB), and other deep neural network models. Using secure encryption algorithms on the data, the model, or both is one solution to preserve data privacy and ensure its sanity within its legitimate environments, especially when used in training or querying large language models. Several prior works have implemented encryption on ML and DL models, securing the training process, the inference pipeline, and the data fed to the model [18–28]. However, when it comes to LLMs to apply homomorphic encryption, the process is sensitive and complex due to the task complexity and the anticipated computational overhead imposed by expensive HE operations. Therefore, attempting to secure an LLM internally within its attention mechanism should be approached carefully, and only the concerned components should be secured to avoid unnecessary overhead while maintaining the desired security level.

In this work, we focus on securing the LLM's inference stage since integrating computationally expensive cryptographic operations into the training process would not be practical and excessively time-consuming. We leverage one of the post-quantum algorithms in fully homomorphic encryption (FHE) to secure the inference process of a specific large language model, Llama-3 [29]. We integrate a specific fully homomorphic encryption scheme offered within the Zama's library,[4] `concrete-ml` [30], which is suitable for the transformers architecture of Llama-3 to run the inference process in fully encrypted mode, hence, secure the data and inference pipeline for selected model's layers. Our approach targets the Multi-Query Attention (MQA) heads of all 32 transformer decoder layers of the `LLaMA-3` model, where the post-quantum cryptographic functions encrypt both the model weights and the attention computations, thereby securing some of the most critical and computationally sensitive components of the inference pipeline. Feed-forward sublayers, layer normalization, and output projection layers remain in plaintext on the server side, while input embeddings are encrypted locally before transmission. The system therefore provides *partial* privacy-preserving inference. This partial encryption is a deliberate design trade-off that makes FHE computationally feasible: encrypting every component of a 32-layer LLM would render inference impractical on current hardware, whereas concentrating FHE on the attention mechanism maximizes the privacy benefit per unit of computational cost. Another motivation for this work is to assess the hardware requirements and costs for such a computational expensive integration.

The rest of the paper is organized as follows: We provide a brief mathematical background about FHE and its main operations, citing the main homomorphic encryption schemes in Section 2, and an overview of `LLAMA-3` model's architecture. Section 3 dissects the transformer's architecture, showing our full FHE integration methodology within its inference components. Section 4 presents the experimental setup and the results of running our homomorphically encrypted model. Finally, we discuss the use cases of our PQC-secured model in Section 5, then conclude the work in Section 6 to draw a comprehensive conclusion about this work.

2. Preliminaries

Fully Homomorphic Encryption is a cryptographic paradigm that enables arbitrary computations on encrypted data without revealing the underlying plaintexts. some FHE schemes rely on the hardness of the *Ring Learning With Errors* (RLWE) problem [31,32]. In RLWE, for a secret polynomial $s \in \mathbb{Z}_q[X]/(X^n+1)$ and a small noise polynomial e, an adversary is given samples $(a_i, b_i = a_i s + e_i) \in (\mathbb{Z}_q[X]/(X^n+1))^2$ and must recover s. The computational intractability of this problem provides the security foundation of modern lattice-based FHE: even if an adversary observes many ciphertexts and intermediate homomorphic computations, recovering plaintexts or secret keys is computationally infeasible, as both a_i and $b_i = a_i s + e_i$ look absolutely random, which makes it hard to distinguish between them, hence, recover the secret s.

Practical FHE implementations, such as those in the `concrete-ml` library, leverage RLWE-based schemes to perform linear transformations (matrix–vector multiplications) and certain nonlinear operations (e.g., approximate ReLU) entirely in the encrypted domain. Techniques such as *ciphertext packing*, *relinearization*, and *bootstrapping* enable parallelism and control noise growth, making it feasible to evaluate large neural networks, including LLMs, without exposing sensitive input embeddings or intermediate weights or activations.

For the rest of the paper, we refer to homomorphic encryption as HE for simplicity, and when providing definitions or mathematical representations, our work depends on a fully homomorphic encryption implementation, referred to as FHE, thereby, both pertain to the same implementation.

[4] An open-source cryptography company pioneering Fully Homomorphic Encryption for confidential applications including AI.

2.1. Mathematical definition

A homomorphic encryption scheme is a tuple of probabilistic polynomial-time algorithms:

$$\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$$

with the following functionalities:

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{ek}, \mathsf{sk})$: Given a security parameter λ, outputs a public key pk and secret key sk, and an evaluation key ek. The latter is used while performing operations on encrypted data, it can be distributed in plain.
- $\mathsf{Enc}_{\mathsf{pk}}(m) \rightarrow c$: Given a message or a plain input $m \in \mathcal{M}$, returns a ciphertext $c \in C$.
- $\mathsf{Dec}_{\mathsf{sk}}(c) \rightarrow m$: Given a ciphertext c and a private key sk, returns the plaintext message m.
- $\mathsf{Eval}_{\mathsf{ek}}(f, c_1, \ldots, c_n) \rightarrow c_f$: Given a function f and ciphertexts $c_1, \ldots, c_n$ corresponding to plaintexts $m_1, \ldots, m_n$, outputs a ciphertext c_f such that:

  $$\mathsf{Dec}_{\mathsf{sk}}(c_f) = f(m_1, \ldots, m_n)$$

Homomorphic encryption bears several properties regarding the operations it supports (addition and multiplication):

Let $\circ$ be a binary operation on plaintext space $\mathcal{M}$ and $*$ the corresponding operation on ciphertexts. A scheme is said to be *homomorphic* with respect to $\circ$ if:

$$\mathsf{Dec}_{\mathsf{sk}}\left(\mathsf{Enc}_{\mathsf{pk}}(m_1) * \mathsf{Enc}_{\mathsf{pk}}(m_2)\right) = m_1 \circ m_2$$

In FHE constructions, noise is an essential security parameter introduced during encryption to mask the underlying plaintext from algebraic recovery. A ciphertext is typically represented as an element c that satisfies a relation such as $c \equiv s \cdot a + e + m \pmod{q}$, where m is the message and e is a small error term. As homomorphic operations are performed, this noise scales according to the complexity of the circuit: for addition, the resulting noise of adding two ciphertexts (c_1 and c_2) is roughly the sum of their respective error values ($e_{add} \approx e_1 + e_2$), whereas for multiplication, the noise grows at a much higher rate, becoming approximately the product of the existing errors ($e_{mult} \approx e_1 \cdot e_2$). If the cumulative magnitude of e exceeds a predetermined threshold dictated by the modulus q, the noise overlaps with the message bits, leading to decryption failures and rendering the data unrecoverable. In this work, we refer to noise by either e or η.

Boolean circuits

Since encryption operations include additions and multiplications, the series of operations can represented in a boolean circuit of **AND**s and **XOR**s within the plaintext space $\mathcal{M} \in \mathbb{Z}_2$ represented in bits and mod 2. Having the homomorphic operations represented in a circuit of universal **NAND** gates (universal since they can represent any logical function), allows estimating a noise budget (how many operations can be performed homomorphically before exceeding the ciphertext correctness threshold on decryption).

Arithmetic circuits

is also an option to represent the operands in integer data types and perform additions and multiplications homomorphically in `modulo` p where p is a prime number greater than 2.

Given the main properties of HE, and their potential circuit representation, it is substantial to distinguish several types of homomorphic encryption approaches based on the operations they support, the multiplicative depth, and their estimated increasing noise:

1. **Partially Homomorphic Encryption (PHE)**: Supports only one operation (either addition or multiplication), such as Rivest–Shamir–Adleman-RSA (multiplicative) and Paillier (additive).
2. **Somewhat Homomorphic Encryption (SHE)**: Supports both operations but only for a limited depth (limited complexity of the f function).
3. **Levelled Fully Homomorphic Encryption (Levelled FHE)**: Supports all operations for functions of bounded depth without noise refreshing.
4. **Fully Homomorphic Encryption (FHE)**: Supports arbitrary depth computations, often achieved by incorporating a *bootstrapping* technique to reduce noise in ciphertexts that increases exponentially after several operations. This operation refreshes the ciphertexts so their decryption would not be corrupted due to uncontrolled noise levels.

2.2. Homomorphic encryption schemes

HE has different schemes applicable to various applications in accordance with the type of operations required for each application. Some perform operations on exact integers $\mathbb{Z}$, while other applications demand operations on approximate real numbers $\mathbb{R}$, such as in our study, since transformers are built on neural networks performing multiplication and addition operations in their network units incorporated within the attention mechanism, in addition to activation, softmax, and other non-linear functions.

Homomorphic Encryption, as a powerful tool for privacy-preserving applications, remained theoretical until Craig Gentry introduced the first FHE scheme in 2009 [33], which enabled the evaluation of arbitrary Boolean circuits on encrypted data. This initial scheme allowed operations such as negation, multiplication, and addition to be performed on ciphertexts without revealing the original plaintexts. Since then, the field has witnessed extensive research and the emergence of further FHE schemes based on diverse hardness assumptions [33–38]. These schemes share critical limitations, including the noise increase due to expensive homomorphic operations within ciphertexts. While addition and multiplication in the binary field F_2 form the basis for Boolean circuit evaluation, they amplify the noise, especially multiplication. This growing noise eventually increases the chances of having incorrect decryption unless avoided by an intermediate operation, bootstrapping. Bootstrapping is a costly procedure for restoring the ciphertext's noise to acceptable levels and preserving ciphertext usability. Due to its computational intensity, only a few FHE schemes have been implemented in practice [39,40], and those suffer from unsatisfactory performance. Furthermore, with each noise mitigation operation comes an additional memory burden as more storage for ciphertext polynomials and/or new matrices for matrix-matrix and vector-matrix multiplication are required. This implies that either CPUs or whatever hardware accelerators (GPUs or FPGAs) are used to run the FHE scheme should have enough vRAM capacity.

Recognizing that many real-world applications do not require arbitrary circuit evaluation, researchers explored alternative designs that forgo bootstrapping. These designs target circuits with low and known multiplicative depth, reducing computational overhead [41–43]. However, schemes such as [39,40,44] face exponential noise growth with circuit depth, significantly limiting their scalability. Brakerski, Gentry, and Vaikuntanathan proposed levelled homomorphic encryption schemes [35], where the noise increases linearly rather than exponentially with the multiplicative depth. Their BGV scheme employs a modulus switching technique and supports plaintexts as polynomials over a ring modulo an integer, enabling batching and larger plaintext spaces. The BGV scheme demonstrated practical use in homomorphic AES evaluation [43], albeit with high resource demands. Specifically, evaluating a depth-d circuit requires storing d variants of an evaluation key, necessitating up to 256 GB of RAM for AES.

Brakerski introduced the concept of scale-invariance [36] at Crypto 2012 in response to the memory and efficiency concerns of modulus switching. Scale-invariant schemes maintain a constant modulus throughout the computation, needing only a single evaluation key copy.

This concept was integrated into BGV by Fan and Vercauteren [45] and into the LTV scheme [37] by Bos, Lauter, Loftus, and Naehrig [42], resulting in the FV and YASHE [42] schemes. While FV remains largely unimplemented, aside from a basic prototype in [25], YASHE became the first operational scale-invariant levelled homomorphic encryption scheme. It achieved promising performance for shallow circuits on personal computers, though it supported only circuits of multiplicative depth up to two due to small modulus constraints.

The `concrete-ml` library utilizes a specialized refresh mechanism called *Programmable Bootstrapping (PBS)* that simultaneously resets the noise level while evaluating non-linear functions through a lookup-table approach. This process takes a noisy ciphertext representing a message m and outputs a refreshed ciphertext representing $f(m)$, where f is an arbitrary function such as a sigmoid or a rectified linear unit (ReLU). Mathematically, while successive operations normally cause the error to grow towards the modulus boundary, this refresh operation maps the current state to a new ciphertext with a fixed, minimal noise term: $c_{out} \approx f(m) + e_{fixed}$. By ensuring that e_{fixed} remains small and independent of the input noise magnitude, the system can execute complex and non-linear machine learning operations of arbitrary depth without the risk of the noise overlapping with the message bits and causing wrong ciphertext decryptions.

2.3. Fully Homomorphic Encryption (FHE) preliminaries

To regroup all potential notations of the main cryptographic operations of FHE, we simulate a minimized representative scenario and all the essential operations regarding data encryption, evaluation, and decryption. In this scenario, a client performs a large language model inference hosted on a non-trusted server while keeping the input data confidential. Let $\mathcal{M}$ denote the plaintext space (e.g., integer embeddings) and $\mathcal{C}$ the ciphertext space. The client generates a public, private, and evaluation keys $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$ and encrypts the input embeddings[5] before sending them to the server.

- **Client [sends]:** Holds sensitive input text and the private key sk, encrypts input embeddings $m \in \mathcal{M}$ using pk, then sends it to the server.
- **Server [receives, computes, sends]:** Hosts the LLM and receives encrypted embeddings and the evaluation key ek, to perform homomorphic evaluation (where main inference operations), including linear layers, attention mechanisms, and activation functions, without accessing plaintext. These phases include bootstrapping and relinearization operations to adjust the increasing noise levels within the resulting ciphertexts.
- **Client [receives]:** Receives the evaluated ciphertexts from the server, then executes the decryption operation using sk to collect the output (model logits).

The general assumption is that the server is honest-but-curious, in other words, it will faithfully execute the model but attempts to infer sensitive information from intermediate ciphertexts. Another assumption which represents attack scenarios is that the server is compromised and the malicious actor controlling it attempts to recover the plaintext(s). FHE ensures that all intermediate activations, gradients, weights, and attention representations remain encrypted with no intermediate decryption operations.

Our approach adopts a *partial* privacy-preserving strategy, where FHE primitives are applied exclusively to the Multi-Query Attention (MQA) heads across all 32 transformer decoder layers, encrypting the Q, K, V, and O projection weights and all associated dot-product computations. The input embeddings (after RMS normalization) are encrypted locally by the client before transmission to the server, ensuring that raw input representations are never exposed in plaintext, while acknowledging that non-attention components remain visible to the server.

The key FHE operations mentioned in this scenario are briefly listed, in operational order, as follows (where $c_i \in C$ denotes ciphertexts in the ciphertext space):

Let $\mathcal{M}$ denote the plaintext space (e.g., integers modulo p) and C denote the ciphertext space. FHE allows computations on encrypted data without accessing plaintext. We summarize these operations below.

- **Key Generation.** The client generates the necessary keys:

  $$\mathsf{KeyGen}(1^{\lambda}) \rightarrow (\mathsf{pk}, \mathsf{ek}, \mathsf{sk})$$

  where λ is the security parameter. For RLWE-based schemes, $\mathsf{sk} \in \mathbb{Z}_q[X]/(X^n + 1)$ and $\mathsf{pk} = (a, b = -a \cdot \mathsf{sk} + e)$ with small error polynomial e.
- **Cryptographic Parameters.** PQC-LLaMA-3 is instantiated with the TFHE/CGGI scheme [46,47] as realized by the `concrete` compiler underlying `concrete-ml`. Ciphertexts are (G)LWE samples over $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$ with a uniform *binary* secret distribution. The compiler's parameter optimizer [48] searches for the cheapest tuple $(N, k, n, \sigma, \beta, \ell)$ that jointly satisfies the circuit's noise-budget (correctness) constraint and a target security level of $\lambda = 128$ bits; candidate tuples below the target are rejected at compile time. Only (N, k, n, q, σ) together with the secret distribution determine λ, whereas the decomposition base and level (β, ℓ) of the bootstrapping and key-switching keys govern correctness and latency but not the underlying (R)LWE hardness. Table 1 therefore reports the complete security-determining parameter set, which is what is required to reproduce the hardness estimate independently.
  For the 8-bit attention circuits evaluated in this work the optimizer returns $k = 1$ and $N = 2048$ (an RLWE dimension of $kN = 2048$), an effective ciphertext modulus of $\log_2 q = 29$ bits, and a Gaussian error with $\sigma/q = 2^{-14.3}$, that is $\log_2 \sigma = 14.7$ in absolute terms. We validated the resulting hardness in two independent ways. First, we evaluated this instance with the `lattice-estimator` [49] under the core-SVP cost model [50], in which a BKZ block size β costs $2^{0.292\beta}$ classically and $2^{0.265\beta}$ quantumly, over the primal-uSVP, primal-BDD (decoding) and dual-hybrid attacks with the binary secret distribution actually used by the compiler. The cheapest attack found is the dual hybrid at block size $\beta = 1394$, costing 2^{407} operations classically and 2^{377} quantumly ($\beta = 1424$), with the primal attacks costlier still (2^{427} classical at $\beta = 1465$), far above the 2^{128} target. Second, we cross-checked the instance against the *Homomorphic Encryption Security Standard* [51], whose tabulation for a ring dimension of 2048 and a discrete Gaussian error of $\sigma = 3.19$ admits $\log_2 q \leq 54$ at 128-bit classical and $\log_2 q \leq 51$ at 128-bit quantum security. Our instance uses $\log_2 q = 29$ with a considerably larger error term, hence it lies well inside the standardized region and retains a substantial margin above the 128-bit post-quantum target rather than sitting at its boundary. Because core-SVP hardness is governed by the dimension and the noise-to-modulus ratio σ/q, the estimate is unchanged when the same instance is expressed at the native 64-bit torus representation of `concrete` ($q = 2^{64}$, $\log_2 \sigma = 49.7$) rather than at the compiler-reported effective modulus; we evaluated both parameterizations and obtained identical attack costs.
  The other evaluated bit-widths (2-bit, 4-bit) change the circuit depth and the noise budget it consumes, and the optimizer re-derives the tuple for each of them under the same enforced $\lambda \geq 128$ constraint. The optimizer is additionally constrained by a bound on the per-lookup-table decryption-failure probability p_{error}, which we leave at the library default. The parameter sets used for every compiled circuit were read back from the compiled artefact (`circuit.complexity_report()`).

[5] The embeddings are the result of feeding the tokenized then encoded raw input into the transformer's embedding layer.

Table 1
Security-determining cryptographic parameter set of the compiled PQC-LLaMA-3 attention circuits. The tuple below is sufficient to reproduce the hardness estimate independently. The decomposition parameters of the bootstrapping and key-switching keys affect correctness and latency only. Estimates use the core-SVP cost model [50] with the `lattice-estimator` [49], the last row gives the corresponding bound of the Homomorphic Encryption Security Standard [51] at the same ring dimension.

Symbol	Description	Value
–	Scheme	TFHE/CGGI (`concrete-ml`)
–	Secret-key distribution	uniform binary $\{0,1\}$
k	GLWE dimension	1
N	Polynomial (ring) dimension	2048
kN	Effective RLWE dimension	2048
$\log_2 q$	Ciphertext modulus	29 bits
σ/q	Relative Gaussian error std. dev.	$2^{-14.3}$
$\log_2 \sigma$	Absolute Gaussian error std. dev.	14.7
λ_{cl}	Classical security, core-SVP $2^{0.292\beta}$	407 bits (dual hybrid, β=1394)
λ_{q}	Quantum security, core-SVP $2^{0.265\beta}$	377 bits (dual hybrid, β=1424)
–	HE-Standard bound at N=2048, σ=3.19	$\log_2 q \leq 54$ (cl.)/51 (q.)

- **Encryption.** A plaintext $m \in \mathcal{M}$ is encrypted as:

  $\mathsf{Enc}_{\mathsf{pk}}(m) \to c$

  In RLWE:

  $c = (c_0, c_1) = (b \cdot r + e_0 + \Delta m, a \cdot r + e_1)$

  where r, e_0, e_1 are small random polynomials and Δ is a scaling factor.
- **Homomorphic Evaluation.** Given $f : \mathcal{M}^k \to \mathcal{M}$:

  $c_{\text{out}} \leftarrow \mathsf{Eval}(f, c_1, \ldots, c_k)$

  Operations include:

  $c_{\text{sum}} = c_a + c_b, \quad \eta_{\text{sum}} \approx \eta_a + \eta_b$

  $c_{\text{prod}} = c_a \cdot c_b, \quad \eta_{\text{prod}} \approx \eta_a + \eta_b + \epsilon q$

  Where η_{sum} and η_{prod} refer to the noise increase in the `Add` and `Mult` operations. The noise amplitude (how large the error term is) increases roughly proportionally to parameters that depend on the polynomial ring size (n) and the ciphertext modulus logarithm $log\ q$.
- **Relinearization.** Reduces ciphertext dimension after multiplications:

  $c_{\text{relin}} \leftarrow \mathsf{Relin}(c_{\text{prod}}, \mathsf{ek})$

- **Programmable Bootstrapping (PBS).** Stands for operations introduced by `concrete-ml` that refresh noisy ciphertexts and prevent them from exceeding the correct decryption threshold

  $c_{\mathsf{fresh}} \leftarrow \mathsf{PBS}(c_{\text{noisy}}, g)$

  where g is a programmable function (identity or non-linear).
- **Decryption.** The client recovers plaintext:

  $$m = \mathsf{Dec}_{\mathsf{sk}}(c) = \left\lfloor \frac{c_0 + c_1 \cdot \mathsf{sk}}{\Delta} \right\rceil \bmod p$$

- **Ciphertext Packing in SIMD.** Multiple ciphertexts can be packed in one global ciphertext to accelerate the homomorphic operations while being executed in a SIMD fashion:

  $c = \mathsf{Enc}([m_1, \ldots, m_\ell])$

  enabling parallel evaluation and amortizing noise growth.

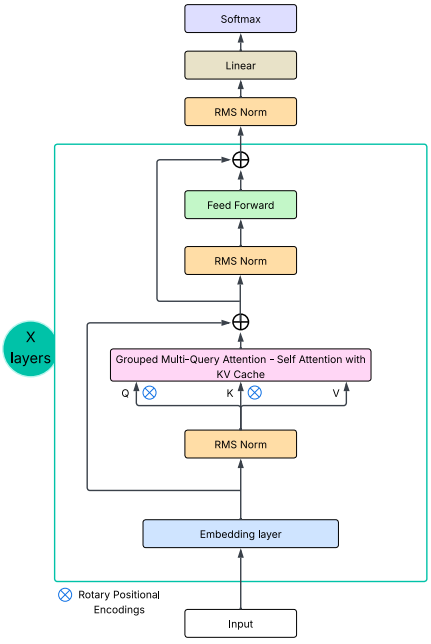


Fig. 1. The LLAMA-3 main architecture.

2.4. LLAMA-3's architecture

The LLAMA-3 model's architecture is based on the transformer decoder's main design with a few changes regarding some internal functions. As Fig. 1 illustrates, the embedding layer takes the raw input and transforms it into a large-dimensional vector (after tokenizing and encoding the input). Nevertheless, the main differences include the used normalization layer (or standardization, which keeps the input's values stable and within a normalized range to ensure a smooth and accelerated model training), where LLAMA-3 uses the Root Mean Square normalization function (RMS). Additionally, the model uses the Rotary Positional Encoding method (RoPE) instead of the standard Positional Encoding method in the original transformer architecture. Concerning memory-related improvements in LLAMA-3 for efficiency, the latter is based on Grouped Multi-Query Attention (GQA) layers to group query heads and share key–value heads within groups without affecting the model's accuracy during the inference phase. The model uses the

Swish Gated Linear Units (SwiGLU) activation function instead of the *Rectified Linear Unit* (ReLU) to add more non-linearity to the model, which allows adaptability towards understanding complex behaviours better than the original function.

Some of the above-mentioned functions will be subject to transformations in LLAMA-3 into a fully homomorphic encrypted inference pipeline, since they are, by nature, computationally expensive and memory demanding compared to linear functions. As HE functions introduce more computations, it is substantial to replace these functions with their efficient function twins in an FHE implementation of the LLAMA-3 model.

In Section 3, we discuss leveraging an existing fully homomorphic encryption library (`concrete-ml`) to integrate its functions within an architecturally modified LLAMA-3 inference model.

3. Methodology

This work will inject homomorphic encryption operations (multiplications, additions, and bootstrapping) from the `concrete-ml` library into the LLaMA-3's transformer architecture, requiring methods overriding and creating new classes inheriting from the mother model (will not be connected to it though under deployment setups).

We build two variants of the encrypted version LLaMA-3, the first one has only its first decoder layer under fully homomorphic encryption settings, while keeping the rest in plain mode. The second variant will have all its layers encrypted. The reason is to measure the impact of FHE operations on the model as whole while tracking its granular impact on a single layer. Moreover, for various applications, encrypting a subset of the model's layers might be considered a good trade-off between privacy and performance given the security and data privacy requirements and restriction bounds.

The new models will be evaluated and tested for text generation accuracy by comparing the match between the generated logits and tokens under the encrypted mode versus the usual plain model text generations. The centre of the model architecture modifications will be within its attention mechanism. Therefore, the attention function should be presented briefly before explaining the core changes in the main LLAMA-3 model functions in this work.

3.1. Attention mechanism

The attention mechanism introduced in [52] has emerged as a pivotal innovation in deep learning, particularly within the domains of natural language processing, computer vision, and time-series modelling. The attention mechanism allows models to dynamically focus on relevant parts of the input sequence when producing the next word (token) while taking into consideration each element of the input. Unlike traditional sequence models such as recurrent neural networks (RNNs), which encode the entire input into a fixed-length vector, attention mechanisms provide a flexible means of accessing the entire input representation at each decoding step. There are multiple variants of the self-attention mechanism, such as Multi-Head Attention (MHA), Multi-Query Attention (MQA), Grouped Multi-Query Attention (GQA) [53], and Sliding Window attention [54].

At its core, the attention mechanism computes a weighted sum of input features (often referred to as values), where the similarity between a query vector and a set of key vectors determines the weights. Formally, given a query Q, a set of keys K, and corresponding values V, the attention output is computed as:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{(d_{emb})}}\right)\mathbf{V} \tag{1}$$

Where d_{emb} is the dimensionality of the keys (K) or similarly, the embedding layer's vector dimension. This formulation, introduced in the seminal Transformer architecture [52], is known as scaled dot-product attention. It enables the model to capture dependencies across different parts of the sequence regardless of their distance, thereby mitigating the limitations of fixed-context or sequential processing as viewed in RNNs.

As demonstrated in Eq. (1), the attention mechanism already comprises a significant amount of multiplications (dot-product & matrix-matrix multiplications) between Q and $K^{\top}$, then scaled by $\sqrt{d_{emb}}$ and finally multiplied with the V matrix after applying the `softmax` function. Even though the amount of multiplication operations is reduced with multi-head or Grouped Multi-Query attention variants, the complexity of the attention process remains expensive.

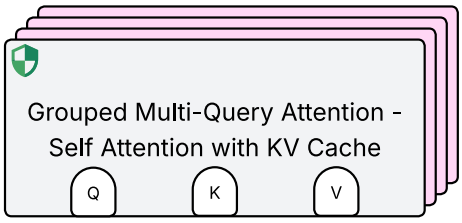
3.2. FHE into LLAMA-3

To introduce FHE in LLMs, specifically to a LLAMA-3 model, we re-implement the inference decoder-based model while integrating `concrete-ml`'s main functions into the attention layers, allowing data encryption and fully homomorphic evaluation of the inference process. Our implementation selectively applies the PQC cryptographic primitives to the attention layer heads, encrypting the associated model weights and performing all attention computations over encrypted data. Therefore, the main operations and transformations within attention heads will be discussed to simplify the understanding process of our changes.

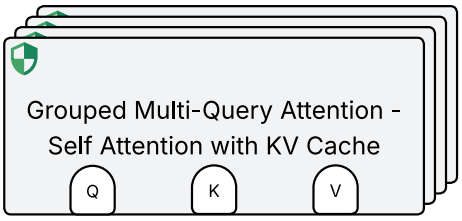
As mentioned previously, LLAMA-3 uses grouped multi-query attention layers to process the inputs through a series of matrix multiplications where the input vectors are multiplied with the model weights to form the Q, K, and V matrices, form the Key-Value (k-v) heads (which takes fewer heads number, for instance, in LLAMA-3 there are 32 heads, the (k-v) grouped heads will be 8), then share them across the Query heads in the grouping process, in which each Key-Value group will be assigned to a query head projection. Next, the query matrices are multiplied by the transpose of key matrices, scaled using the square root of the embedding dimension, and then finally, multiplied by the value matrices after applying the softmax function. This operation (attention calculation) is repeated for each layer in the Llama model. We transform the LLAMA-3 architecture into a fully homomorphically encrypted (FHE)-compatible model for inference through quantization and selective FHE-aware computation. The central aim of this approach is to adapt the attention mechanism in the LLAMA-3 model such that a subset of the computations, especially the most sensitive and computationally expensive functions, can be executed securely over encrypted data using the `concrete-ml` library. To this end, we introduce two key architectural modifications (see Fig. 3): the Quantized Llama Single Head Attention (`QLlamaSingleHeadAttention`) and the Quantized Llama Multi-Head Model (`QLlamaLMHeadModel`) classes, collectively enabling grouped multi-query encrypted attention heads within a LLAMA-3 modified inference model.

3.3. Quantized attention with FHE integration

At the core of the proposed methodology is the re-engineering of the Llama attention layers. Standard Llama uses a multi-head attention mechanism with high-precision floating-point matrix operations. The `concrete-ml` library does not fully support multiplication and addition of floating-point numbers. Therefore, we leveraged the model integer quantization within the library on the modified LLaMA-3 model and ran it in FHE-compatible functions. Consequently, we name our new fhe-supported and Quantized LLAMA-3 model the `PQC-LLaMA-3` model. The quantized attention replaces the original attention heads with fixed-point arithmetic within this model, where the encryption scheme is applied to both the attention layer weights and the computational operations performed on these weights during inference. Specifically:



(a) An abstract view of the developed `QLlamaSingleHeadAttention` implementing the main QLlamaAttention that overrides the original LLAMA-3 self-Attention layers. The figure shows a single homomorphically encrypted grouped multi-query attention head



(b) An abstract view of the developed `QLlama3MultiHeadsAttention` implementing the main QLlamaAttention that overrides the original LLAMA-3 self-Attention layers. The figure shows a a multi-head homomorphically encrypted grouped multi-query attention heads. It is the result of concatenating multiple encrypted single attention heads.

Fig. 2. Abstract views of the modified Grouped Multi-Query Attention component, showing the `QLlamaSingleHeadAttention` and the `QLlama3MultiHeadsAttention` variants.

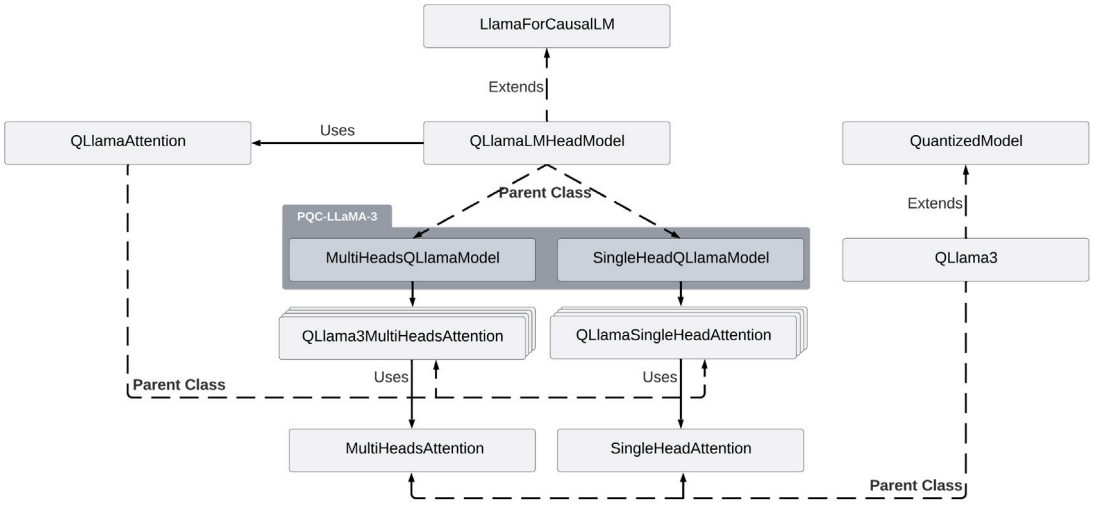


Fig. 3. Class hierarchy of the proposed FHE-enabled QLLaMA-3 models. Encrypted attention modules inherit from the original LLaMA-3 architecture while selectively replacing attention heads with quantized and FHE-compatible counterparts. The classes in darker grey represents the main executable model variants of `PQC-LLAMA-3`.

1. Projection Replacement: In the SingleHeadAttention class illustrated in Figs. 3 and 2(a), the initial linear transformations that generate the Query, Key, and Value vectors—originally computed via high-dimensional matmul over the attention weights (`c_attn`)—are replaced with quantized linear projections implemented using the DualArray abstraction (a proposed data structure used in `concrete-ml` library), maintaining both floating-point and integer representations.
2. Selective Head Quantization & Encryption: Rather than performing FHE computations across all 32 heads of LLAMA-3, we isolate a single attention head (typically the first) for encrypted evaluation (attention mechanism computations), implemented via `QLlamaSingleHeadAttention`. This balances between the desired security and model's text generation efficiency for certain applications that counts a single layer encryption sufficient to provide reasonable privacy levels for the input data and the model weights.
3. Attention Computation Replacement: The softmax-scaled dot-product attention (i.e., $Softmax(QK^T/\sqrt{d_{emb}})\times V$) is restructured in the quantized attention module. The computation is decomposed into FHE-compatible operations: encrypted dot products for QK^T, normalization via polynomial approximations (replacing the softmax function with a FHE-compatible approximation function), and secure matrix–vector multiplications for the attention output.
4. FHE Execution Modes: To facilitate FHE integration at different development stages, the `QLlamaAttention` class introduces a FHE mode toggle. In `disable` mode, the model retains its quantized weights and integer arithmetic but evaluates clear tensors without encryption (quantization-only baseline). In `simulate` mode, the Concrete runtime emulates the numerical behaviour of the compiled FHE circuit without performing ciphertext execution. In `execute` mode, inputs are encrypted and the compiled circuit performs actual homomorphic evaluation. Accordingly, the three-way baseline uses `disable` for the quantized-only configuration and `execute` for the quantized-plus-FHE configuration.

3.4. LLAMA-3 *model modification for quantized integration*

The `QLlamaLMHeadModel` class extends `LlamaForCausalLM`, substituting the attention module in a designated transformer layer with our quantized version. We localize encryption overhead and maintain compatibility with downstream decoder components by targeting a specific layer (e.g., the first).

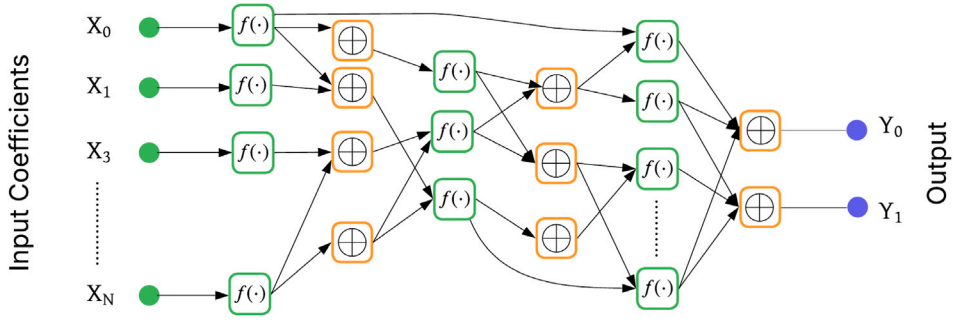


Fig. 4. Programmable Bootstrapping Evaluation Circuit formed of univariate and linear functions. It reduces noise per function (Evaluation + Bootstrapping)

Prior to FHE compilation, the model performs a cleartext calibration pass to determine the quantization parameters required for encrypted execution. This calibration step estimates scaling factors, zero-points, and value ranges for each quantized operation, which are stored using `concrete-ml`'s DualArray structure. These parameters ensure numerical consistency between cleartext and encrypted inference, and define the fixed-point representation that will be used within the FHE circuit.

Following calibration, the encrypted computation graph is generated, as illustrated in Fig. 4, by compiling the forward pass of the attention module using the `concrete-ml` compilation API. During this phase, the FHE circuit structure is fully determined, including the placement and configuration of PBS operations. Although PBS is not executed at compile time, its count, location, and cryptographic parameters (e.g., key-switching and precision constraints) are statically inferred from the circuit graph. In particular, PBS nodes are inserted wherever nonlinear operations or noise-refresh steps are required to maintain correctness under FHE constraints. As a result, the compiled circuit encodes a noise-safe execution plan, while the actual noise management and bootstrapping are performed during encrypted execution, transparently to the user.

The described architecture is modular, allowing for straightforward substitution of additional layers or expansion to multi-head encrypted attention. In our implementation, in addition to `QLlamaSingleHeadAttention`, which encrypts only one head, the framework supports generalization to multiple heads through multiple duplications of the developed `SingleHeadAttention` structure, resulting in the `QLlama3MultiHeadsAttention` (See Fig. 2(b) compared to Fig. 2(a)).

3.5. Homomorphic evaluation and Llama inference

After the compilation step where the general FHE circuit for the attention mechanism is built, the input text (under its embedding format) is encrypted into ciphertexts using the previously generated public key (compilation phase) on the client side. Furthermore, linear layers (projections such as `q_proj`, `k_proj`, `v_proj`, and `o_proj`), quantized activations, and intermediate integer arithmetic are translated into low-level cryptographic operations, mainly additions, ciphertext multiplications, and PBS operations (for noise refresh). Once the input is encrypted and the evaluation keys are generated, they are passed to the server side to run the expensive attention layer(s) mechanism operations in FHE mode. The server executes the circuit, producing encrypted outputs, which the client then decrypts to recover integer representations. It would result in a correct decyrption for two reasons: ***i***) the public, private, and evaluation keys were generated by the authenticated and legitimate client, ***ii***) the PBS operations ensure that the ciphertexts' noise level does not exceed the correct decryption threshold. The result decryption in integer format is then dequantized back to floating numbers.

The decrypted logits (or outputs) are passed to softmax layer (not the attention softmax function, check Fig. 5) in clear, decode token IDs into words to complete the model's auto regressive text generation task.[6]

The `SingleHeadQLlamaModel` and `MutliHeadsQLlamaModel` represent the `PQC-LLaMA-3` model we propose in this work, they are the main variants in which we observe the effect of encrypting one transformer layer (and its attention heads) or all layers on the overall model's performance, including inference time and tokens/second metrics. The `SingleHeadQLlamaModel` model does not mean running it with only one layer, instead, encrypting one layer, while keep the rest (31 layers) in plain.

3.6. Threat model: Partial privacy-preserving

We consider an *honest-but-curious* server adversary, where the server faithfully executes the inference protocol as specified but attempts to infer private information from all intermediate values it can observe during the forward pass. The server is assumed to be computationally bounded and is modelled as a probabilistic polynomial-time (PPT) adversary. It may log, store, and analyse any plaintext activations it receives, yet it cannot break the underlying FHE scheme, whose security reduces to the hardness of the (RLWE) problem [31].

The server has full read access to (i) the ciphertext transmitted by the client, (ii) all intermediate activations produced by non-attention sublayers, namely the feed-forward sublayers, RMS layer normalizations, and the output projection, and (iii) the final output logits. All of these components execute in plaintext on the server (Fig. 5). The server does *not* observe plaintext token embeddings or any intermediate representation processed inside an FHE-encrypted attention block, as those values remain encrypted throughout computation.

Prior to transmission, the client locally encrypts the post-embedding, post-RMS-normalization input representations under the FHE public key. Decryption requires the FHE secret key, which never leaves the client. Any attempt by the server to recover plaintext values from the received ciphertext is therefore computationally infeasible under the RLWE hardness assumption.

Because neither `SingleHeadQLlamaModel` nor `MultiHeadsQLlamaModel` encrypts the full model, activations that flow through plaintext non-attention sublayers are visible to the server. Specifically, the residual stream entering and exiting each feed-forward sublayer, the outputs of every RMS normalization, and the final hidden states fed into the language-model head are all observable. This constitutes a *partial* privacy guarantee, the confidentiality is ensured only at the attention-computation boundary. Accordingly, the property we establish is the confidentiality of the client's input representation and of

[6] Autoregressive generation: feed predicted token(s) back for next token generation.

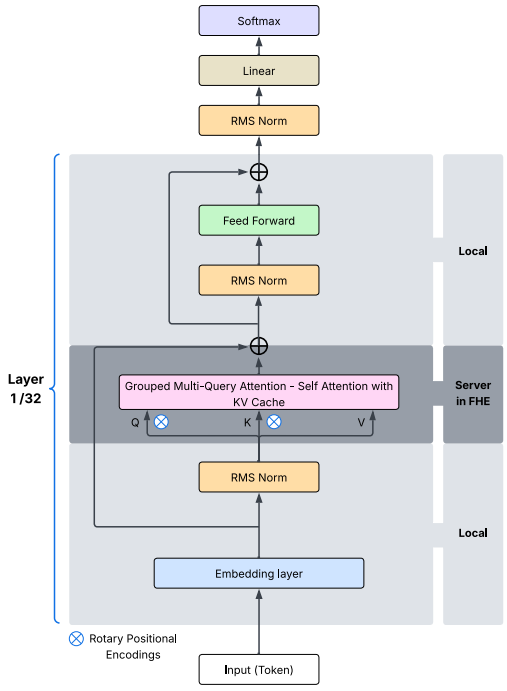


Fig. 5. An overview of the PQC-LLaMA-3 model for inference. It processes the input until the first layer locally, then sends it to the server side to homomorphically evaluate the attention mechanism. The evaluation result is sent back to the local machine to continue the inference process. This is a model describing an efficient use case of our model within a client–server model where the server has powerful hardware to run the expensive homomorphic operations on the attention mechanism.

the attention computation, and not indistinguishability of the prompt given the server's complete plaintext view of the feed forward model component, where the latter would require encrypting the remaining sub-layers and is explicitly out of scope for this work (ongoing future work). The resulting privacy exposure and the information an honest-but-curious server could extract from these plaintext activations are characterized in detail in Section 5.1. It is important to note that the next near-future work incorporates including all non-linear functions such as the softmax, GELU (Gaussian Error Linear Unit), and LayerNorm functions in the FHE space within the LLaMA-3 models.

4. Experimental setup and results

We measure the performance of our developed PQC-LLaMA-3 model by comparing its text generation with a plain LLaMA-3 model. Therefore, we selected a set of 79 text prompts and feed it to each model (plain and encrypted) and for `SingleHeadQLlamaModel` and `MultiHeadsQLlamaModel` models, then incentivize it to complete the inserted prompts (unfinished sentences and paragraphs) by generating the potential next words (tokens). Furthermore, we vary the number of new tokens to generate for both encrypted model variants (*single* and *multi-head*) and repeat them 5 times, then report their average.

The main metrics to evaluate in this work are the correctness of the large language model text generation in a fully homomorphic encrypted modified and quantized Llama-3 inference model. Hence, we assess the effect on the model's accuracy after converting floating numbers to integers and run the attention mechanism arithmetic operations in FHE mode. The other metrics are the execution times of the inference and compile processes, in addition to the tokens per second readings of our proposed model compared to the plain Llama-3 model. We used the machines described in Table 2 for the designed trials then compared their performances.

Table 2
Hardware Specifications of both machines, running on an Ubuntu 22.04.4 LTS operating system.

Node	Count	CPU	Storage	Cache
machine 1 (M1)	1	AMD EPYC 7413 24-Cores (2.65 GHz)	4.5TB SSD	L1d: 1.5 MiB L1i: 1.5 MiB L2: 24 MiB L3: 256 MiB
machine 2 (M2)	1	12th Gen Intel(R) Core(TM) i7-12700 K (3.6 GHz	1TB SSD	L1: 1.0 MiB L2: 12.0 MiB L3: 25.0 MiB
machine 3 (M3)	1	14th Gen Intel(R) Core(TM) i9-14900 K (3.2 GHz)	2TB	L1d: 896 KiB L1i: 1.3 MiB L2: 32 MiB L3: 36 MiB

4.1. Token-agreement rate

The *generation accuracy* reported in Tables 3–5 is a *top-k token-agreement rate*, it represents the fraction of prompts for which the plain model's argmax next-token falls within the top-k candidates produced by the partially encrypted model. In other words, it measures the degree of matching between the plain model's greedy token selection and the FHE model's top-k output set, quantifying distributional consistency between the two models rather than correctness against any external ground truth. Because a wider sampling window trivially increases the probability of capturing the reference token, the agreement rate grows monotonically with k regardless of encryption fidelity. To complement this metric, Section 4.8 provides distribution-independent quality indicators (perplexity, BLEU-4, ROUGE-L, GLUE, and BERTScore) that assess intrinsic text quality without dependence on the sampling-window size.

The experimental evaluation is organized into two main categories related to the length of the generated new tokens (context length). The first category considers evaluating the PQC-LLaMA-3 model when generating from 1 to 10 new tokens, which is considered short-context generation and simulates applications such as classification/labelling or short question answering. The second category illustrates the model's generation for a longer context (from 70 to 500 new tokens) fitting applications such as report generation, classification with explanation, and text summarization.

We measure the impact of the quantization (quantization set to 2 bits since it was observed to achieve the best performance) and fully homomorphic encryption operations on the generation accuracy of PQC-LLaMA-3 and its average inference time, then draw a few conclusions about the model's performance for short and long-text generation. Each category covers the two trial modes offered by `concrete-ml`, the *simulate* and *execute* modes. The obtained results are further discussed in the appendices section, where they are linked to the cache memory access patterns and the different processors' capacities.

4.2. Short-context generation

Short-context token generation, limited to 1–10 new tokens, stresses the model's ability to produce *immediate outputs* when there is little temporal averaging. For a single step t, the logits are computed as

$$\mathbf{z}_t = W_o \mathbf{h}_t, \tag{2}$$

where $\mathbf{h}_t$ is the Transformer hidden state and W_o is the output projection. Under FHE, each inner product in $\mathbf{z}_t$ is evaluated on ciphertexts c_i with inherent noise η_i. The effective noise level per token is therefore

$$\eta_{t_j} \approx \frac{\|W_o \mathbf{h}_t\|}{\|\eta_i\|}. \tag{3}$$

Table 3
Generation Accuracy (%)/Inference Time (s) for n_bits=2 integer quantization on Three Machines for the `SingleHeadQLlamaModel` running in two modes (`simulate/execute`).

Top-k	Trial	Accuracy	(M1)	(M2)	(M3)	Compile time (s) (M1/M2/M3)
1	Simulate	47.2	0.472	0.264	0.239	8.02/4.420/3.497
	Execute		0.572	0.266	0.237	8.02/4.455/**3.471**
2	Simulate	61.0	0.490	0.261	0.241	8.01/4.412/3.574
	Execute		0.474	0.261	0.241	7.60/4.342/3.565
3	Simulate	67.4	0.526	0.262	0.241	7.9/4.427/3.627
	Execute		0.588	0.262	0.24	8.02/4.328/3.572
4	Simulate	70.6	0.492	0.262	0.241	7.58/4.422/3.640
	Execute		0.523	0.261	0.236	7.9/4.377/3.625
5	Simulate	73.1	0.521	0.261	0.242	7.82/4.437/3.638
	Execute		0.525	0.261	0.24	8.06/4.353/3.604
6	Simulate	75.9	0.541	0.261	0.238	7.72/4.402/3.571
	Execute		0.615	0.262	0.24	8.10/4.300/3.57
7	Simulate	77.6	0.527	0.261	0.238	7.61/4.429/3.651
	Execute		0.549	0.263	0.239	8.06/4.332/3.653
8	Simulate	79.2	0.542	0.263	0.242	7.63/4.494/3.633
	Execute		0.540	0.262	0.238	7.58/4.377/3.616
9	Simulate	80.9	0.529	0.262	0.239	7.82/4.449/3.614
	Execute		0.594	0.262	0.237	7.94/4.352/3.62
10	Simulate	**81.7**	0.558	0.261	0.238	7.95/4.401/3.616
	Execute		0.568	0.261	0.239	**8.555**/4.350/3.618

Table 4
Generation Accuracy (%)/Inference Time (s) for n_bits=2 integer quantization on Three Machines for the `MultiHeadsQLlamaModel` running in two modes (`simulate/execute`).

Top-k	Trial	Accuracy	(M1)	(M2)	(M3)	Compile time (s) (M1/M2/M3)
1	Simulate	51.6	1.575	0.909	0.835	26.190/14.882/12.286
	Execute		1.558	0.902	1.055	26.369/14.667/**12.254**
2	Simulate	66.4	1.625	0.893	0.948	26.138/14.868/12.737
	Execute		1.618	0.886	1.142	26.295/14.592/12.391
3	Simulate	72.9	1.542	0.929	0.942	26.083/15.040/12.780
	Execute		1.640	0.899	1.239	26.295/14.673/12.785
4	Simulate	76.9	1.601	0.941	0.954	26.095/15.148/12.685
	Execute		1.532	0.883	1.241	26.178/14.549/12.832
5	Simulate	80.0	1.652	0.985	0.958	26.238/15.219/12.278
	Execute		1.641	1.08	1.219	26.264/15.957/13.044
6	Simulate	82.2	1.653	0.967	0.958	26.323/15.330/12.502
	Execute		1.616	1.0	1.220	26.213/15.589/12.958
7	Simulate	83.6	1.554	0.965	0.953	26.083/15.425/12.496
	Execute		1.631	0.905	1.232	26.108/15.656/12.834
8	Simulate	85.8	1.600	0.939	0.947	26.284/15.252/12.794
	Execute		1.616	0.970	1.259	26.170/15.608/12.733
9	Simulate	86.8	1.629	0.957	0.956	**26.343**/15.407/12.809
	Execute		1.599	0.905	1.244	26.329/15.056/12.773
10	Simulate	**87.4**	1.679	0.932	0.956	26.088/15.321/12.554
	Execute		1.635	0.904	1.261	26.295/15.083/12.779

Where t_j represents token j from the full set of generated tokens. Because only a few tokens are generated, the model cannot benefit from the statistical smoothing of attention over many steps; even minor quantization or decryption noise directly perturbs $\mathrm{softmax}(\mathbf{z}_t)$,[7] making this regime a stringent test of per-step numerical fidelity.

Tables 3 and 4 illustrate the multiple runs of the `SingleHeadQLlamaModel` and `MutliHeadsQLlamaModel` models for top-$k \in [1, 10]$ tokens across all available fhe modes.

[7] The softmax function converts the generated logits from the previous steps at a specific layer into a probability distribution, indicating the model's confidence that the input belongs to a specific class or that a particular word is the most likely next word

4.2.1. Single-head FHE-model performance

The results for the `SingleHeadQLlamaModel` are summarized in Table 3. Across all machines, the generation accuracy steadily increased with higher top-k values, starting from 47.2% for $k = 1$ to 81.7% for $k = 10$, following the expected top-k sampling trend. Interestingly, quantization did not degrade accuracy as simulated and executed modes preserved identical generation scores as the unquantized baseline, confirming the correctness of the 2-bit integer quantization scheme.

From a performance standpoint, inference times on M3 were consistently the lowest across all settings, with average inference latencies, for `simulate/execute` modes, around 0.24/0.238s compared to 0.261/0.262s on M2 and 0.519/0.554s on M1. These latencies specifically measure the computational cost of the encrypted attention layer heads, encompassing both the encrypted model weights and the homomorphic

operations performed on the attention mechanism. This reflects the superior single-thread performance of the Intel i9-14900 K compared to the other processors given their turbo frequencies, 6 GHz for M3, 5 GHz for M2, 3.6 GHz for M1, respectively.

Compilation times showed a similar pattern, decreasing from 8.55 s on M1 to 3.47 s on M3 (see the row in bold in Table 3). Only the maximum values of the compiling times were considered.

This reduction aligns with the faster memory access and higher turbo frequencies of consumer CPUs, suggesting that CPU frequency and memory latency are stronger determinants of compilation speed than total threads or system RAM size.

Comparing the quantization modes, both "`simulate`" and "`execute`" introduced negligible runtime overhead (under 5%) relative to the unquantized baseline. This indicates that integer-based quantized operations were efficiently integrated within the CPU execution pipeline, confirming the low-bit quantization can be CPU-friendly with minimal computational penalty.

4.2.2. Multi-head FHE-model performance

The `MultiHeadsQLlamaModel` results, presented in Table 4, follow similar accuracy trends compared to Table 3, but with overall higher performance. Accuracy improved from 51.6% ($k = 1$) to 87.4% ($k = 10$), confirming that the multi-head configuration better captures contextual variance even under low-bit quantization. However, the additional attention heads increased computation time. Average inference latency (`simulate/execute`) rose to approximately 1.611/1.60s on M1, 0.941/0.936s on M2, and 0.94/1.211s on M3.

Since all computations were performed on CPUs, this increase stems from the higher volume of matrix multiplications and attention aggregation steps required per inference cycle. Nevertheless, the latency growth, compared to the single-head quantized model, remained sublinear relative to the number of heads, showing that multi-threaded CPU execution effectively distributed the operations across cores.

Compilation times for the multi-head model were notably longer, the maximum timing reached 26.343s on M1, while the minimum compilation time was took 12.254s on M3, due to the additional complexity in generating optimized kernel graphs for multi-head attention paths. Once again, compile times correlated strongly with CPU clock frequency and memory access speed, reaffirming that CPU-bound quantized models remain sensitive to low-level architectural characteristics such as cache size and instruction-level parallelism. These and subsequent results are discussed in depth in Appendix B.

4.3. Long-context generation

In long-context generation the model autoregressively (where the generation of the next token depends also on the previously generated ones) produces up to 500 tokens, repeatedly updating each model's decoder layer hidden states. This setting evaluates the *cumulative stability* of the encrypted pipeline, the ciphertext noise effect along with homomorphic operations controlling error across layers. We evaluate the models within the long-context generation for both PQC-LLaMA-3 model's variants, the `SingleHeadQLlamaModel` and `MultiHeadsQLlamaModel` models.

4.3.1. Single-head FHE model performance

For the `SingleHeadQLlamaModel`, token-agreement rate improved monotonically from 91.3% at 70 new tokens to 98.2% at 500 new tokens (see Table 5). The increase tracks the generation length. Over a longer autoregressive span the agreement statistic is accumulated across many more steps, which smooths isolated single-step mismatches, and ciphertext packing together with periodic key switching stabilizes the encrypted arithmetic over long spans (Section 4.4.2), so per-step noise averages out rather than compounding.

The results reveal a clear inverse relationship between CPU frequency and latency, with M3 achieving the lowest times. The inference time differences between M2 and M3 (~10%), even though CPU frequency-wise, M2 should be performing better than M3, can be attributed to the L2/L3 cache size and memory-controller efficiency, rather than to thread count. M1's higher latency mainly stems from its lower per-core frequency, which introduces additional data-access delays in FHE encryption/decryption cycles. The stability of inference times (standard deviation < 5%) also suggests that FHE-enabled quantized operators were efficiently optimized for CPU cache hierarchies, achieving near-deterministic performance. Overall, the `SingleHeadQLlamaModel` demonstrates that FHE-encrypted inference can achieve sub-second latency on modern CPUs.

Table 5
Generation Accuracy (%)/Inference Time (s) for n_bits=2 quantization on Three Machines for `SingleHeadQLlamaModel/MultiHeads-QLlamaModel` models running in the FHE mode (`execute`).

Top-k	Accuracy	M1	M2	M3
70	91.3/94.7	0.556/1.553	0.256/1.478	0.233/0.790
80	91.7/94.8	0.501/1.545	0.258/1.403	0.237/0.879
90	92.1/94.9	0.508/1.547	0.256/1.433	0.235/0.969
100	92.3/95.1	0.482/1.591	0.257/1.260	0.235/0.938
110	92.4/95.1	0.478/1.643	0.257/1.154	0.237/0.841
120	92.8/95.3	0.464/1.646	0.261/1.248	0.234/0.926
300	96.2/97.1	0.456/1.633	0.268/1.762	0.235/1.02
400	96.5/97.4	0.487/1.721	0.270/1.899	0.236/0.928
500	**98.2/97.7**	0.516/1.652	0.276/2.070	**0.236/0.825**

4.3.2. Multi-head FHE model performance

As Table 5 shows, the `MultiHeadsQLlamaModel` displayed similar accuracy trends, increasing from 94.7% at 70 new tokens to ~ 97.7% at 500 new tokens. This improvement confirms that the multi-head structure enhances contextual representation, even under a series of expensive FHE operational mode.

As expected, the introduction of multiple attention heads increased computational cost. On M1, inference time varied between 1.55 s and 1.72 s, while M2 recorded values between 1.15 s and 2.07 s, and M3 achieved the lowest latency between 0.79 s and 1.02 s. The results again demonstrate that inference time is influenced, in addition to the FHE operations cost, by per-core clock frequency and cache hierarchy efficiency. The higher variance observed on M2 at larger generation lengths indicates occasional cache saturation or operating system context-switching overheads, as multiple encrypted tensors compete for limited L2 and sometimes a generous L3 cache space.

Compared to the single-head variant, the multi-head model incurred an average latency increase of approximately 3.5× on M1, 4.3× on M2, and 4.3× on M3. This moderate increase demonstrates that quantized multi-head FHE inference remains computationally feasible on high-frequency CPUs, especially when parallelized across available cores. The latency scaling was sublinear relative to the number of attention heads, confirming that the parallelization of encrypted operations effectively mitigates the expected computational overhead.

4.4. Analysis of long-versus short-context encrypted generation

The objective of these experiments is to examine how fully homomorphic encryption and quantization affect text generation performance in the PQC-LLaMA-3 model under two distinct regimes: (i) *short-context* generation of 1–10 new tokens, and (ii) *long-context* generation, extending up to 500 new tokens. Although both settings employ identical model weights, encryption parameters, and keys, they reveal different operational sensitivities. The Short-context emphasizes the immediate impact of encryption and quantization, including ciphertext noise on each prediction step, while long-context generation probes the cumulative noise behaviour and the stability of encrypted arithmetic over extended autoregressive sequences.

Two main factors affect the PQC-LLaMA-3 variants (SingleHeadQLlamaModel and MultiHeadsQLlamaModel) performance are the noise introduced by encrypted data and operations, in addition to the ciphertext packing mechanism in concrete-ml library. Their contribution to the performance (positively or negatively) is discussed briefly in the next two subsections.

4.4.1. Noise under encrypted operations

The base architecture of LLaMA-3 is a decoder-only Transformer with L layers ($L = 32$), each containing GQA self-attention, feed-forward sublayers, layer normalization, and residual connections. At decoding step t, the hidden representation $\mathbf{h}_t$ is updated as in the standard Transformer formulation, and token probabilities are obtained via

$$P(x_t \mid x_{<t}) = \mathrm{softmax}(W_o \mathbf{h}_t), \tag{4}$$

where W_o denotes the output projection matrix, and the softmax represents the decoder layer's final function. In the PQC-LLaMA-3 model, these floating-point operations are replaced by fixed-precision arithmetic over ciphertexts encrypted in the RLWE domain, introducing *quantization* and *homomorphic noise*.

Quantization error arises from mapping real/floating point-valued tensors to low-bit-width integer representations prior to encryption (using 2-bit quantization since concrete-ml support only integer arithmetic operations). Model parameters and intermediate activations are scaled and rounded according to calibrated quantization parameters, ensuring compatibility with homomorphic evaluation while bounding numerical distortion. This error is deterministic and fixed once the quantization scheme is selected.

Homomorphic noise, by contrast, is an inherent property of RLWE-based fully homomorphic encryption schemes. Each ciphertext embeds a noise component that guarantees semantic security (see Sub- Section 2.1). During encrypted inference, this noise grows with every homomorphic operation—most notably ciphertext–ciphertext multiplications used in dot-product computations within the attention mechanism. Additions incur negligible noise growth, while multiplications result in multiplicative noise expansion proportional to the ciphertext modulus and operand magnitudes.

These noise factors decreases the encrypted model's accuracy especially when the noise levels exceeds the calculated budget, leading to wrong decryptions of homomorphically calculated attention values, which would consequently, lead to wrong next token generation.

To prevent uncontrolled noise accumulation and ciphertext invalidation, the PQC-LLaMA-3 compilation process explicitly inserts PBS operations at strategically selected points in the encrypted attention circuit. PBS simultaneously refreshes ciphertexts by reducing accumulated noise and enables the evaluation of non-linear functions (e.g., activation and non-linear functions approximations) in the encrypted domain. Importantly, the placement and frequency of PBS operations are determined statically during circuit compilation using the library's framework, based on worst-case noise growth analysis of the computation graph.

4.4.2. Ciphertext packing

In the FHE setup, ciphertext packing embeds multiple ciphertext slots into a single ciphertext via the Chinese Remainder Theorem (CRT) for batching, and Number Theoretic Transforms (NTT) for efficient polynomial arithmetic, enabling SIMD-style parallel and fast matrix multiplications within the attention and feed-forward layers. This design amortizes noise across slots and reduces per-slot variance, effectively improving the arithmetic precision of batched operations. Following ciphertext–ciphertext multiplications, key switching (or relinearization) re-expresses results under the same secret key and controls modulus expansion, mitigating excessive noise accumulation (which is achieved by the PBS operations).

These mechanisms ensure the numerical stability of multi-layer encrypted computation, particularly over longer generation spans, which interprets the slightly increasing accuracy for the chosen top-k values (see Table 5) even with the presence of FHE operations and the augmenting noise effects.

Empirically, short-context generation exhibits a sharper sensitivity to quantization precision and single-step ciphertext noise, as predictions depend heavily on the most recent encrypted hidden state. This results in a lower effective precision ratio (EPR)[8] at each decoding step:

$$\mathrm{EPR}_{\mathrm{short}} \sim \frac{\|W_o \mathbf{h}_t\|}{\|\eta_{\mathrm{short}}\|}, \tag{5}$$

where η_{short} denotes the noise component in the encrypted logits.

In contrast, during long-context generation, ciphertext packing and periodic key switching distribute and stabilize noise across multiple tokens, leading to a higher effective EPR:

$$\mathrm{EPR}_{\mathrm{long}} \sim \frac{\|W_o \mathbf{H}\|}{\|\eta_{\mathrm{long}}\|_{\mathrm{eff}}}, \tag{6}$$

where $\mathbf{H}$ collects the hidden states of all generated tokens. Experimentally, this behaviour manifests as improved generation coherence and reduced degradation in perplexity for sequences up to 500 tokens. The results suggest that noise averaging through ciphertext packing and recurrent key switching contributes to better overall numerical stability, explaining why the PQC-LLaMA-3 model maintains higher accuracy during long-context decoding despite fully homomorphic constraints.

4.5. The compilation bottleneck

As observed in Tables 3 and 4, the model compilation process to generate the FHE circuit forms a bottleneck for this work since the operation execution times exceeds 8 secs and 26 secs for the SingeHeadQLlamaModel and MultiHeadsQLlamaModel models, respectively. While it appears to be an unsatisfying result where a user would not possibly wait for multiple seconds to have a text generated, nevertheless, this is avoidable by simply constructing the model's FHE circuit only once, then apply it on the ensemble of prompts/queries the users intend to submit. The circuit can be changed or re-constructed in case the task's nature changes or when the prompts' structure, type, and length vary due to multiple querying scenarios.

The potential re-generation of the FHE circuit is enabled by the flexibility of the developed PQC-LLaMA-3 model in this work, it offers model compilation as callable function at any desired code segment or model inference routine.

4.6. The KV-Cache effect

As noticed in Tables 3, 4, and 5, the average inference times are almost the same regardless of the number of new tokens to generate. It seems logical to expect rising inference times whenever the model generates more text, however, this performance is valid and achievable thanks to the *KV-caching* technique implemented in LLaMA-3 models, and consequently, inherited in our PQC-LLaMA-3 model as well. This option enables the auto regressive model to avoid re-computing the previous sequence of tokens preceding the one to be generated. It is an expensive computational operation to perform the same operations repeatedly, therefore, *KV-Caching* technique allows the model to store the previously generated sequence in the cache and fetch it for future generations.

To delve deeper into investigating this specific execution and memory behaviour of the model, we collected microarchitectural performance metrics using the Linux perf tool to quantify the execution behaviour of each model variant (single and multi heads models)

[8] This metric is not meant to be reported in the experiments, it is used exclusively in this section.

across different generation-length configurations. To ensure consistency and reduce noise, each experiment was executed on a CPU core pinned to a single logical processor using `taskset`, avoiding thread migration.

Hardware performance counters were recorded using `perf stat` with a fixed set of events (their names vary depending on the processor and its kernel architecture), including `cycles`, `instructions`, `L1-dcache-loads`, `L1-dcache-load-misses`, `L1-icache-loads`, `L1-icache-load-misses`, `L2_RQSTS.REFERENCES`, `L2_RQSTS.MISS`, `LLC-loads`, `LLC-load-misses`. Multiple runs (10 trials per configuration) were averaged to account for system variability, and the collected counters were subsequently used to observe the cache memory accesses on all levels which explains the model's behaviour (similar inference times for all generation lengths). Appendix B illustrates the results from the collected cache memory accesses, for both `SingeHeadQLlamaModel` and `MultiHeadsQLlamaModel` models, and then discusses further the enhanced memory access nature in LLaMA-3 models which also manifests in the PQC-LLaMA-3 encrypted model.

4.7. FHE-related metrics

We propose a set of new metrics to evaluate the efficiency of the `PQC-LLaMA-3` model including the number of executed PBS operations, the average of PBS count per generated token, the average needed memory per token (in bytes), and the overall throughput for both `PQC-LLaMA-3`'s variants, `SingleHeadQLlamaModel` and `MultiHeadsQLlamaModel`.

Under this FHE-secured `LLaMA-3` implementation, a sequence length of 6 tokens (for text completion or generation) requires 5208 PBS operations where the implementation reported in [30] requires 11622 for a FHE `GPT-2` model. Despite that both models have different attention head architectures, it still counts as a relatively more efficient implementation compared to the mentioned prior work.

We introduce some new metrics recorded from the model's evaluation trials, illustrated in Table 6. In order to capture the following metrics, we run trials varying the number of new tokens to generate within the range [10–500] and for each value, we run the models on the 79 text prompts then average the recorded metrics, except for the PBS_count which is fixed for each model.

4.7.1. Average throughput

The tokens per second is used as the model's throughput metric to evaluate the model under plain and FHE-encrypted settings, and observe the effect of homomorphic encryption integrated functions on the model's autoregressive token generation performance.

$$\text{Throughput} = \frac{\text{Total Tokens Generated}}{\text{Total Execution Time}} \quad \text{[tokens/second]} \tag{7}$$

The Total Execution Time corresponds to the mean end-to-end execution time per inference run. This normalized formulation mitigates variability across runs and provides a stable indicator of sustained token generation performance. In the FHE setting, throughput reflects both the cost of homomorphic arithmetic and the overhead introduced by cryptographic primitives such as programmable bootstrapping. Table 6 represents our encrypted models throughput (token/sec), PBS operations per generated token, and average used memory per token.

4.7.2. Average PBS per token

The average PBS per token measures the cryptographic intensity of encrypted inference. It is computed as

$$\text{Avg_PBS_Token} = \frac{\text{PBS_Count}}{\text{Generated_Tokens}}, \tag{8}$$

and represents the mean number of programmable bootstrapping operations required to generate a single token. Since PBS operations dominate latency and energy consumption in RLWE-based FHE schemes, this metric provides a direct measure of the homomorphic cost per output token. Importantly, this quantity depends on the encrypted circuit structure—such as the number of attention heads, sequence length, and non-linear layers and functions—and is independent of decoding parameters such as *top-k*.

4.7.3. Average memory per token

The average memory per token, expressed in bytes, captures the memory overhead induced by encrypted inference. It is defined as

$$\text{Avg_Mem_Token} = \frac{\text{Total_Mem}}{\text{Generated_Tokens}}. \tag{9}$$

This metric accounts for the storage required by encrypted attention layers, weights, intermediate ciphertexts, and bootstrapping artefacts throughout the inference process. Lower values indicate better memory efficiency and improved scalability, which are critical for deployment on resource-constrained hardware.

The results shown in Table 6 demonstrate a decent average throughput for the `SingleHeadQLlamaModel` and an efficient distribution of PBS operations per generated token, while consuming 2.02 KBytes of main RAM memory. As for `MultiHeadsQLlamaModel`, its average throughput is low compared to the single FHE-head model, reflected by the high average PBS operations per generated token (13067.62) which explains the low throughput, since a higher PBS per token would cause the model to perform poorly increasing the latency of its text generation. However, the memory consumption is nearly the half of what `SingleHeadQLlamaModel` consumes. It is substantial to note that the models use the cache memory and its levels besides the main memory when storing the packed ciphertexts, attention matrices, non-linear activations and gradients.

4.8. Quantization and quality analysis

This subsection presents complementary experiments that isolate the effects of quantization from FHE encryption, provide standard NLP quality metrics, and analyse the sensitivity of the token-agreement rate to the sampling-window size.

We evaluate our FHE implementations using the instruction-tuned LLaMA variant, LLaMA-3.2-3B-Instruct model, because it produces task-focused, constrained outputs that reduce stylistic variance between the plain and encrypted models, enabling more meaningful quality comparison. Instruction-tuned models generate better-controlled outputs than base models, whose unconstrained continuations exhibit divergence dominated by sampling randomness rather than FHE effects, making it difficult to attribute quality differences to the encryption pipeline.

In the following experiments, we chose a larger set of 200 prompts curated for the instruct LLaMA models, different from the previous set used in the token-agreement experiments.

4.8.1. Evaluation metrics

We adopt perplexity, BERTScore-F1 [55], BLEU-4 [56], and ROUGE-L [57] as primary quality indicators. Perplexity quantifies the model's predictive confidence on its own continuations (lower is better), while BERTScore captures semantic similarity via contextual embeddings, offering robustness to the lexical divergence inherent in autoregressive generation. BLEU-4 measures the precision of four-gram overlaps between generated and reference text, providing a surface-level fidelity signal that is particularly sensitive to exact phrasing and fluency. ROUGE-L complements this by measuring the longest common subsequence between output and reference, rewarding recall of salient content regardless of contiguous n-gram matches. We retain BLEU-4 and ROUGE-L as primary metrics because, unlike general instruction-following settings, our evaluation compares FHE-generated outputs against a fixed plain FP16 reference. In this context, n-gram deviations are a direct signal of quality degradation caused by the encryption scheme.

Table 6
Encrypted inference efficiency metrics for single-head and multi-head PQC-LLaMA-3 models on machines M1 and M3 given as averages for each PQC-LLaMA-3 model. We refer to SingleHeadQLlamaModel as Single and MultiHeadsQLlamaModel as Multi.

Model	M1			M3		
	Throughput	PBS/token	Mem (byte)/token	Throughput	PBS/token	Mem (byte)/token
Single	355.30	60.51	18952.67	77.26	137.62	2019.96
Multi	**16.67**	9604.25	597.46	8.91	13067.62	1119.11

GLUE is a collection of natural-language-understanding tasks designed to evaluate transferable language representations [58]. Our evaluation covers SST-2 for binary sentiment classification, QNLI and RTE for textual-entailment classification, and MRPC for paraphrase identification. For each task, exact-match classification accuracy is the fraction of generated labels that equal the task's gold labels. When a single GLUE score is reported, it is the unweighted macro-average of the four task accuracies, so each task contributes equally regardless of its evaluated subset size. These benchmark accuracies measure correctness against external ground-truth labels and are therefore distinct from the token-agreement rate, which measures only consistency between plain and partially encrypted generations.

4.8.2. Evaluation protocol, repetitions, and reported uncertainty

All generation experiments use deterministic greedy decoding with a fixed prompt ordering and fixed evaluation subsets. The plain FP16 and quantization-only configurations are therefore reproducible up to a low run-to-run variance ($\leq$5%). The encrypted configurations are not deterministic, meaning that TFHE encryption is randomized, so every execution draws a fresh key set and fresh encryption noise and a repeated run may decrypt to a different integer, and hence to a different token. Each encrypted generation measurement reported in Section 4 is consequently the mean of five independent repetitions, each performed with a freshly generated key set and independent encryption randomness, where the relative standard deviation of the encrypted latency stays below 5% across machines and generation settings. The quality metrics of this section aggregate 200 prompts $\times$ 50 new tokens per configuration, that is on the order of 10^4 encrypted token steps, so the reported means are averages over a large sample rather than single-run measurements.

The present work includes the GLUE suite as its general language-model benchmark, with the four tasks reported in Table 9 evaluated on fixed 200-item subsets of their public validation splits. Broader public suites (MMLU, HellaSwag, LAMBADA, and WikiText-2 perplexity) are planned as future work, as they fall outside the main scope of tests for this phase of the work.

4.8.3. Three-way baseline comparison

To isolate the performance impact of quantization from FHE encryption, we evaluate three configurations on the same prompt set using the multi-head architecture: (a) plain LLaMA-3 in FP16, (b) the same multi-head architecture with quantization only in `disable` mode, and (c) the quantized architecture with FHE in `execute` mode. The comparison is controlled such that configurations (b) and (c) share identical architecture, weights, and quantization parameters, and only clear integer evaluation versus ciphertext execution differs. Table 7 summarizes the three-way comparison. The quantization-only overhead (b vs. a) isolates the cost of integer arithmetic, while the additional FHE overhead (c vs. b) reflects the cost of actual homomorphic evaluation.

To avoid self-reference bias, all three configurations are benchmarked against an external reference corpus of 200 standard answers manually curated to match the difficulty and length distribution of the instruct prompts. This external reference approach directly measures semantic preservation relative to independent ground-truth answers rather than relative to the plain model's own outputs. This approach enables us to interpret configurations (b) and (c) not as relative degradations from (a), but as absolute semantic-fidelity measurements anchored to ground-truth expectations.

Table 7
Three-way baseline comparison (Multi-head architecture, 200 prompts, 50 new tokens). BERTScore-F1 measures semantic similarity against an external reference corpus of 200 verified standard answers, avoiding self-reference bias if compared to the FP16 model. Configuration (a) Plain FP16 achieves 86.4% against external reference, configuration (b) Quant-only 86.1% (0.3pp degradation from quantization), and configuration (c) Quant+FHE achieves 84.41% (an additional 1.7pp degradation compared to (b) from FHE applied on top of quantization). Quality values are reported as mean $\pm$ standard deviation over the repetitions described in Section 4.8.2.

Configuration	Bits	FHE	BERTScore-F1 (vs ext. ref.)
(a) Plain FP16	FP16	disable	86.4 $\pm$ 1.0
(b) Quant only	8-bit	disable	86.1 $\pm$ 0.9
(c) Quant + FHE	8-bit	execute	**84.4** $\pm$ 1.2

The results reveal that BERTScore-F1 is marginally affected by FHE encryption, configuration (c) achieves 84.41 compared to configuration (b)'s 86.1 (1.97% drop), indicating strong semantic preservation under FHE. Almost a negligible drop rate stems from quantization, as the configuration (b) achieves 0.4% percentage drop compared to original model (a), while FHE adds a 2.31% total degradation from the plain model. This demonstrates that quantization has a minor effect on the quality loss, with FHE contributing a major additional effect on semantic similarity when applied to the attention heads.

4.8.4. Quantization bit-width ablation

Another substantial experiment is to measure the effect of quantization precision independently of FHE noise, therefore, we evaluate all three bit-widths (2-bit, 4-bit, 8-bit) with FHE disabled and compare perplexity, BLEU-4, ROUGE-L, and BERTScore-F1 against the same external reference corpus used in Section 4.8, with ΔPPL reported relative to the FP16 row. Table 8 reports the results. For LLaMA-3.2-3B-Instruct, the quantization ablation reveals non-monotonic quality degradation across bit-widths, where 8-bit quantization achieves the best quality under with perplexity 1.28 (ΔPPL = +0.10), BLEU-4 of 40.3, ROUGE-L of 54.1, and BERTScore-F1 of 86.1%, closely preserving near-FP16 baseline performance. 2-bit shows moderate degradation (PPL 1.3, ΔPPL = +0.12, BLEU-4 38.9, ROUGE-L 53.8, BERTScore 78.2%), while 4-bit exhibits anomalously poor performance, indicating a quantization-specific instability in the 4-bit precision regime for this model size. The FHE model row combines quantization and FHE effects, comparing against 8-bit only, FHE adds approximately +0.81 PPL increase, illustrating the cumulative impact of both factors on downstream inference quality.

4.8.5. Controlled diagnosis of the 4-bit anomaly

The 4-bit result is not a uniform loss of quality but a dissociation between metric families, and this dissociation is itself diagnostic. Read against the 2-bit row of Table 8, the 4-bit configuration loses well over half of BLEU-4 (15.6 against 38.9) and of ROUGE-L (31.6 against 53.8), yet scores higher on BERTScore-F1 (79.8% against 78.2%), while perplexity rises to 2.31. BERTScore is computed from contextual embeddings and is insensitive to surface realization, while BLEU-4 and ROUGE-L are *n*-gram precision and longest-common-subsequence

Table 8
Quantization bit-width ablation (Multi-head architecture, 200 prompts, 50 new tokens). PPL = perplexity is computed on continuation tokens only. BLEU-4, ROUGE-L and BERTScore-F1 are measured against the external reference corpus of 200 verified standard answers used in Table 7, ΔPPL is relative to the FP16 row. Values are mean ± standard deviation over the repetitions of Section 4.8.2.

Model	Precision	PPL	ΔPPL vs FP16	BLEU-4	ROUGE-L	BERTScore-F1
LLaMA-3.2-3B-Instruct	FP16 (reference)	**1.18** ± 0.06	0.00	**45.8** ± 1.9	**56.2** ± 2.0	**86.4** ± 3.1
	2-bit	1.3 ± 0.04	+0.12	38.9 ± 1.4	53.8 ± 1.7	78.2 ± 2.5
	4-bit	2.31 ± 0.06	+1.13	15.6 ± 0.4	31.6 ± 0.9	79.8 ± 2.9
	8-bit	1.28 ± 0.03	+0.10	40.3 ± 1.2	54.1 ± 1.4	86.1 ± 2.7
	FHE Model	**2.09** ± 0.07	+0.91	**35.6** ± 1.0	**51.8** ± 1.7	**84.4** ± 3.5

Table 9
GLUE benchmark accuracy across plain FP16 and quantized partial-FHE (8-bit) configurations. Evaluated on 200 samples per task with deterministic decoding. The average represents an unweighted macro-average across all four tasks. Accuracies are mean ± standard deviation over the repetitions of Section 4.8.2.

Task	Plain FP16 (%)	FHE 8-bit (%)	Accuracy Drop (pp)	Category
SST-2	90.0 ± 1.1	78.5 ± 2.7	−11.5	Sentiment
MRPC	43.5 ± 1.1	42.0 ± 1.3	−1.5	Paraphrase
RTE	66.5 ± 1.2	54.0 ± 1.4	−12.5	Entailment
QNLI	68.5 ± 3.1	44.0 ± 1.7	−24.5	Entailment
Average	**67.1** ± 1.7	**54.6** ± 1.8	**−12.5**	**All Tasks**

measures. The pattern therefore localizes the anomaly to the surface form of the generations, that is length inflation, reordering, and local repetition, rather than to a collapse of the internal representation, where the 4-bit model still produces semantically related content but stops reproducing the reference phrasing. A uniform loss of representational quality cannot produce this ordering, since it would move the embedding-based indicator in the same direction as the *n*-gram ones, whereas 4-bit is markedly worse than 2-bit on the latter and slightly better on the former.

The most plausible mechanism is the interaction between the per-tensor symmetric quantizer used by `concrete-ml` and the small number of high-magnitude outlier channels that are known to dominate the activation range of LLaMA-class models [59,60]. Because the calibration-derived scale is fixed by those outliers, the resolution effectively available to the remaining channels does not vary monotonically with the nominal bit-width, which is consistent with an intermediate precision behaving worse than both of its neighbours. Attributing the effect conclusively requires controlled ablations that vary one factor at a time, specifically per-tensor versus per-channel scales, calibration-set size and composition, and the outlier-clipping percentile, each followed by a full re-quantization and re-evaluation of the 200-prompt set and repeated over several seeds. Those runs form part of our ongoing work on quantization-aware FHE compilation and are not reported here. The evidence available at present is nonetheless sufficient to exclude semantic collapse as the explanation, and we accordingly retain 8-bit as the reference precision for all quality claims in this paper, while 2-bit is used where circuit depth and PBS count dominate the latency budget.

4.8.6. NLP quality and downstream benchmark assessment

Table 9 complements the generation-quality measures (PPL, BLEU-4, ROUGE-L, and BERTScore) with ground-truth downstream accuracy on the selected GLUE tasks. The comparison uses identical prompts, label verbalizations, deterministic decoding, and evaluated subsets for the plain FP16 and quantized partial-FHE models. Consequently, the observed differences isolate model-configuration effects rather than sampling variation.

The downstream results reveal differentiated accuracy retention across GLUE tasks. SST-2 retains 78.5% accuracy, a −11.5 percentage-point drop from the plain FP16 baseline of 90.0%, indicating reasonable semantic preservation for sentiment understanding. MRPC shows minimal degradation (−1.5pp, from 43.5% to 42.0%). RTE metric exhibits moderate degradation (−12.5pp, from 66.5% to 54.0%), showing substantial FHE-induced quality loss on entailment reasoning. QNLI exhibits the largest accuracy loss (−24.5pp, from 68.5% to 44.0%), reflecting the task's sensitivity to subtle semantic distinctions that are more prone to quantization error. These figures are obtained zero-shot with a strict exact-match verbalizer, which places the absolute FP16 baselines well below fine-tuned levels (and, for MRPC, below majority class). Table 9 should therefore be read as relative utility retention under identical prompting for the two configurations, not as absolute task performance. Overall, averaging across all four tasks yields plain FP16 accuracy of 67.1% and FHE 8-bit quantized accuracy of 54.6% (12.5 pp average drop), demonstrating that the partial-FHE model preserves utility for shallow semantic tasks (sentiment, paraphrase) while incurring substantial quality loss on deep reasoning and entailment tasks.

5. PQC-LLaMA-3 vs. attacks on LLMs

We provide in Table 10 a set of attacks on AI models and LLMs, mostly targeting decrypted outputs and weights acquisition to perform a successful attack. Nevertheless, it is substantial to set a clear assumption about the environment where our developed model would best fit. Therefore, we assume the following while providing the model's FHE protection assessments: The cryptographic security of the system relies on the standard hardness assumptions underlying lattice-based FHE schemes (e.g., Learning With Errors). Encryption and decryption are performed by a trusted client holding the public and private keys, while the evaluation key is delivered to the server (where evaluation takes place). The FHE server operates deterministically without malicious code injection or output tampering. Model weights and quantized tensors are pre-encrypted or protected in secure memory before deployment. Output decryption and interpretation occur in a trusted environment isolated from the adversary.

Even though fully homomorphic encryption is quantum-resistant, it is not immune to all types of attacks targeting machine learning, deep learning, or large language models. There are always risks and potential security flaws surrounding the data and model's privacy, which are a direct result of bad software implementations, hardware implementations, or both. Moreover, there exists a set of attacks that target even FHE-encrypted models where the encryption cannot prevent their risks and consequences on data and model privacy (Table 10 discusses a few of them).

Table 10
Assessment of FHE protections for common attacks in LLM deployment.

Attack	FHE-protected	Explanation
Prompt/context leakage	Yes	The prompt and all attention-internal state are encrypted, so the server never observes the input text, its embeddings, or the attention computation.
Embedding inversion (prompt reconstruction from input representations)	Partial	The client encrypts the post-embedding representation before transmission, so the server never observes the plaintext input embeddings on which inversion attacks operate to recover the prompt verbatim [61–63]. However, the plaintext input to the first feed-forward sublayer equals the sum of the (encrypted) input embedding and the first attention output, and therefore correlates with the embedding; inversion mounted against this and the deeper plaintext residual states is not excluded.
Attention-state probing (Q/K/V and attention-map inference)	Partial	Query, key, value tensors, attention scores, and the projection matrices remain ciphertexts under RLWE throughout the encrypted decoder layers, blocking token- and attribute-level inference that relies on access to internal attention state [64]. The attention output itself, however, is still recoverable (Section 5.1).
Output reconstruction (decryption guessing)	Yes	Properly configured FHE ciphertexts do not reveal plaintext, but implementation errors, malleability, or metadata/timing side channels can leak information.
Gradient leakage (reconstruction)	No	FHE protects gradients/activations *in-flight* (ciphertexts). If gradients or aggregated updates are revealed in plaintext, or if protocol steps require decryption, leakage remains possible. Hybrid/training protocols may introduce leakage points.
Data poisoning	Partial	Encryption does not prevent an authorized adversary from submitting poisoned data; while obscuring inputs can hinder targeted poisoning, hardening the attempts to poison data and complicates dataset auditing, however, validating inputs and queries submission authentication is required to achieve a better security.
Side-channel attacks (timing, cache, EM)	No	FHE does not inherently remove physical side-channels. Constant-time implementations, hardware isolation (TEEs), and side-channel hardening remain necessary.
Model extraction (stealing)	No	FHE hides internals from an entity that only observes ciphertexts, but black-box extraction via plaintext query outputs is still possible unless outputs are restricted (DP, clipping, rate limits).
Adversarial example injection	No/Partial	FHE prevents internal gradient access, but black-box adversarial attacks (query-based) remain feasible.

5.1. Privacy boundaries of partial-FHE inference

In the context of this work, *privacy* denotes the semantic security of attention-layer weights, query/key/value/output projections, and all intermediate dot-product computations under the (RLWE) hardness assumption. We define the precise security boundary of the PQC-LLaMA-3 construction as follows.

- **Protected components.** The attention projection matrices (W_Q, W_K, W_V, W_O) and all associated dot-product computations in the encrypted decoder layer(s) are secured under RLWE-hardness. An honest-but-curious server observing the encrypted attention block cannot recover these weights or the intermediate query, key, and value tensors. In the `MultiHeadsQLlamaModel` variant, all 32 decoder layers' attention computations are encrypted, providing a substantially stronger barrier against weight and activation extraction.
- **Unprotected components.** For both variants, feed-forward sublayer activations, layer normalization outputs, and output logits are computed in plaintext on the server. Importantly, input token embeddings (after RMS normalization) are not exposed in plaintext to the server since they are encrypted locally by the client using the FHE public key before transmission, as depicted in Fig. 5. The residual-stream structure of the transformer means the server can access the hidden state $\mathbf{h}_\ell$ leaving each encrypted attention block (in plaintext form after the attention output is added to the residual). Because both the residual stream entering a feed-forward sublayer and the one leaving the previous sublayer are visible, the server recovers each encrypted attention block's output *exactly*, as the difference of adjacent plaintext residual states, for every layer after the first; no probing or estimation is required. At the first layer this differencing is blocked (the input embedding is encrypted), but the plaintext feed-forward input equals the embedding plus the first attention output and therefore correlates with the embedding. What remains protected are the attention weights and the internal Q/K/V computations, which stay semantically secure under RLWE: the server holds the block's inputs and outputs but cannot read projection matrices or per-token attention scores out of the ciphertexts. Functional extraction from the observed input–output pairs is a separate, non-cryptographic risk and falls under the model-extraction row of Table 10, which FHE alone does not prevent.
- **Single-head vs. multi-head security gap.** In the `SingleHeadQLlamaModel` (1 of 32 layers encrypted), the server observes 31 decoder layers and all feed-forward blocks in plaintext, substantially limiting the privacy guarantee. The `MultiHeadsQLlamaModel` (all 32 attention blocks encrypted) provides a much stronger barrier, where the server sees only feed-forward sublayers and the output projection in plaintext, with all attention-level computations and input embeddings protected. The aim for this work is to provide a partial privacy-preserving implementation on computationally expensive transformer architectures and fully homomorphic encryption primitives, it minimizes the set of possible attacks on plain models, limiting attackers to only sophisticated options targeting the model's open feed forward, normalization, and output logits components.
- **Scope of the privacy claim.** We deliberately do *not* claim end-to-end privacy. What the construction establishes is an *input-side and attention-side* confidentiality guarantee: the prompt never

Table 11
Quantitative comparison of privacy-preserving inference approaches for transformer models. GLUE-Avg = average accuracy across reported GLUE tasks; PPL = Perplexity; N/R = not reported in original work; N/A = not applicable. PQ = post-quantum security. Measurement type: *Emp.* = empirical on reported hardware; *Lit.* = values taken from the original publication without hardware standardization. Latency figures are heterogeneous: per-token for generative models, per-sentence/sample for encoder models. GLUE scores are on development/validation splits where reported; averages cover only the tasks evaluated in the original work.

Method	Approach	Privacy scope	Latency	GLUE-Avg (%)	PPL	Security	Type
PQC-LLaMA-3 (ours, multi-head, 3B)	FHE (TFHE)	Attention (32 layers)	825 ms/token	~54.6[a]	2.09	128-bit PQ	Emp.
IRON [65]	HE + MPC (2PC)	Full model (BERT-Tiny)	~2.5 min/sample[b]	~74.2[c]	N/A	128-bit	Lit.
CrypTFlow2 [66]	2PC	Full DNN (ResNet-50)	~ 545.8 s LAN/ 936 s WAN[d]	N/A[e]	N/A	128-bit	Lit.
MPCFormer [67]	MPC (2PC)	Full model (BERT-Base)	~26.8/sequence	~84.0	N/R	128-bit	Lit.
THE-X [68]	FHE (CKKS)	Full model (BERT-Tiny)	N/R	<1.5% drop[f]	N/R	128-bit	Lit.
BumbleBee [69]	HE + 2PC	Full model (BERT-Base/LLaMA-7B)	~8 min/token (7B)[g]	≈plaintext[h]	N/R	128-bit	Lit.
BOLT [70]	HE + MPC (2PC)	Full model (BERT-Base)	~185 s (LAN)–1744 s (WAN)[i]	≈plaintext[j]	N/R	128-bit	Lit.
Concrete-ML [30]/GPT-2 (1.5B)	FHE (TFHE)	Full model (GPT-2)	~300 s/token[k]	N/A	N/A	128-bit PQ	Lit.

[a] Estimated from reported task scores (SST-2 78.5%, MRPC 42.0%, RTE 54.0%, QNLI 44.0%), empirical on machines M1/M2/M3.
[b] IRON reports end-to-end runtime on BERT-Tiny under LAN (377 MBps, 0.8 ms) and ~2.5 min per sample, BERT-Base would be substantially longer (~100 min per the SIRNN comparison in Figure 4 of the original paper).
[c] Accuracy loss <0.3% vs. plaintext on SST-2, MRPC, MNLI, QNLI with BERT-Tiny.
[d] Semi-honest 2PC on ResNet-50/ImageNet under LAN.
[e] Computer-vision benchmark only, ImageNet Top-1 accuracy bitwise-equivalent to plaintext.
[f] THE-X reports <1.5% performance drop on BERT-Tiny across downstream tasks, absolute GLUE scores not aggregated in the paper, we calculated their average and it is 71.82.
[g] Published as NDSS 2025, LLaMA-7B token latency ≈ 8 min on CPU.
[h] BERT-Base accuracy matches plaintext on SST-2, MRPC, QNLI, CoLA (Table III of paper).
[i] End-to-end BERT-Base inference, LAN ≈ 185 s with word elimination. WAN (100 Mbps, 80 ms) ≈ 1744 s (Table 3/Fig. 5 of paper).
[j] Maximum accuracy loss 1.3% vs. floating-point plaintext on MRPC, RTE, SST-2, STS-B.
[k] Per-token latency estimated from reported throughput.

leaves the client in plaintext, and every query, key, value tensor, attention score, and projection matrix of the encrypted decoder layers is semantically secure under RLWE. This boundary is sufficient to exclude the class of adversaries that operate on the input representation or on internal attention state, namely embedding-inversion and generative embedding-inversion attacks that reconstruct the prompt from token embeddings [61–63], attribute and membership inference from intermediate representations [64], and attention-map probing, the corresponding rows of Table 10 are marked *partial* accordingly: the internal attention state is protected while the block outputs are not. It is not sufficient to exclude reconstruction from the plaintext tensors that remain, namely the residual stream leaving each encrypted attention block, the feed-forward activations, the normalization outputs, and the final logits. Bounding the information recoverable from those tensors, for instance by mounting inversion probes against $\mathbf{h}_\ell$ and reporting an achieved reconstruction rate, requires a dedicated attack study that we did not perform here, it is being carried out alongside the extension of homomorphic coverage to the non-linear sub-layers announced in Section 6. Until that evidence exists, the correct reading of our result is a quantified reduction of the leakage surface with respect to a fully plaintext pipeline, and not a proof of end-to-end privacy, providing a trade-off between latency and privacy given the expensive computation budget of fully and homomorphically encrypted models.

- **Formal security considerations.** A complete formal reduction of this partial-FHE construction to an established RLWE hardness assumption (analogous to full-circuit FHE security proofs) is beyond the scope of this work and is noted as a direction for near-future investigation and development. What we demonstrate is that the attention weights and computations in the encrypted layers are individually semantically secure under RLWE, and that the overall scheme provides measurably stronger protection against prompt extraction, weight theft, and gradient leakage than a fully plaintext pipeline. Complementary security mechanisms such as identity protection frameworks for distributed systems [71], differential privacy, output clipping, and trusted execution environments can further strengthen the overall deployment security when combined with FHE-protected attention layers, while keeping reasonable inference latencies.
- **Comparison with alternative privacy-preserving approaches.** Table 11 provides a quantitative comparison of PQC-LLaMA-3 against published privacy-preserving inference approaches, standardized to the inference latency, GLUE-Avg accuracy and perplexity where available. The table reports latency in heterogeneous units (ms/sec/min) to reflect literature reporting practices, with detailed footnotes explaining model-size differences (ranging from 0.1M params for BERT-tiny to 7B params for LLaMA-7B) and measurement types (empirical vs. literature-derived).
  Our approach is distinguished by two primary quantitative advantages: (i) it achieves 825 ms/token latency on commodity CPUs with post-quantum security (RLWE-based TFHE, 128-bit), while maintaining GLUE-Avg ≈54.6% and perplexity 2.09 on 3B-parameter LLaMA. (ii) It is the only method combining post-quantum security without server-side hardware trust and generation-oriented NLP metrics (PPL, BLEU, ROUGE, BERTScore) on billion-scale decoder-only models. The quantitative comparison shows the following: IRON (~2.5 min/sample, GLUE ≈74.2%) and MPCFormer (~26.8 s/sequence, GLUE ≈84.0%) leverage

seconds-to-minutes latency with partial post-quantum guarantees on BERT models. BumbleBee (480 s/token on 7B LLaMA, ≈plaintext) combines HE+2PC but incurs extreme latency. THE-X applies CKKS-FHE to BERT-tiny (0.1M params, GLUE ≈71.8%) with no reported latency. Concrete-ML (FHE on GPT-2, 1.5B params) achieves ∼300 s/token but lacks GLUE evaluation. CrypTFlow2 (545.8–936 s) targets vision models and lacks NLP metrics. Critically, existing approaches either omit perplexity measurements or evaluate on encoder-only models without generation-quality metrics. Our partial-FHE approach on LLaMA-3.2-3B achieves the fastest latency (825 ms/token) among post-quantum-secure methods with measurable generation quality (PPL 2.09, BLEU-4 35.6%, ROUGE-L 51.8%, BERTScore-F1 84.4%) on a billion-scale decoder-only LLM.

Our work is among a few to evaluate FHE-based inference using generation-oriented metrics on a decoder-only LLM at the billion-parameter scale with post-quantum security, providing a competitive latency among PQ-secure methods with quantified privacy-latency-quality trade-offs.

5.2. Scalability considerations

The experimental results reveal a clear scalability trade-off in the PQC-LLaMA-3 framework. Encrypting all 32 attention blocks (the `MultiHeadsQLlamaModel` variant) is feasible on current hardware but increases the PBS count per token by approximately 95× compared to the single-head variant (from ∼60 to ∼9,604 PBS/token on the machine `M1`), with a corresponding 3–8× latency increase per generated token. Full-layer encryption would require circuit depths that exceed current `concrete-ml` capabilities and would likely render interactive inference impractical without substantial hardware acceleration (e.g., Accelerated GPU FHE operations including bootstrapping, dedicated FHE ASICs or FPGA accelerators).

Regarding context length, the KV-caching mechanism inherited from LLaMA-3 ensures that inference latency scales linearly with the number of new tokens rather than quadratically, as demonstrated by the near-constant per-token latency observed in Tables 3–5. Extending to larger models (e.g., 70B parameters) would require distributed FHE across multiple nodes, with ciphertext communication overhead becoming a dominant bottleneck. Larger batch sizes are constrained by the ciphertext expansion factor, which increases memory requirements by approximately 50–100× compared to plaintext tensors. These scalability challenges motivate hybrid deployment architectures where FHE-protected attention is combined with complementary privacy mechanisms (TEEs for feed-forward layers, differential privacy for output perturbation) to achieve comprehensive coverage without prohibitive overhead.

Given this assessment, we highly recommend the coupling of this proposed secure version PQC-LLaMA-3 model with a set of other security mechanisms to reinforce the overall model's security and data privacy against the mentioned attacks. The security mechanisms comprise input sanitization and output validation, differential privacy, output clipping, query limits, and establishing trusted execution environments.

6. Conclusion

We introduced PQC-LLaMA-3, a partially privacy-preserving inference pipeline that integrates TFHE-based fully homomorphic encryption into the attention mechanism of all 32 decoder layers of LLaMA-3, while retaining plaintext feed-forward, normalization, and output projection stages. This design targets a practical operating point in the privacy–utility–efficiency trade-off space, providing post-quantum confidentiality for the attention computation without requiring server-side trusted hardware or interactive multi-party protocols.

The empirical results substantiate the viability of this design. On LLaMA-3.2-3B-Instruct, the partial-FHE model attains BERTScore-F1 of 84.4%, perplexity of 2.09, and BLEU-4/ROUGE-L of 35.6%/51.8% against an external reference corpus, indicating that the encrypted attention path preserves the semantic content of generated text. Our three-way baseline isolates the sources of quality degradation and shows that FHE contributes only a ∼ 1.7 pp additional drop beyond quantization, so ciphertext arithmetic itself is not the dominant cost of accuracy. The bit-width ablation (2/4/8-bit) further confirms that 8-bit quantization remains close to FP16 quality (PPL 1.28), while lower precisions incur task-dependent degradation. Downstream, GLUE evaluation exhibits the expected task-sensitivity gradient: shallow semantic tasks such as SST-2 (78.5%) and MRPC (42.0%) degrade modestly, whereas entailment-heavy QNLI (44.0%) is more affected. In the broader context of privacy-preserving inference, our method achieves 825 ms/token, the lowest latency among post-quantum-secure approaches to date, demonstrating that FHE-protected attention on billion-parameter decoder-only LLMs is attainable on commodity CPUs.

These findings must be qualified by clear limitations. Encryption is currently restricted to the attention sub-layers, however, the residual stream, MLP blocks, layer normalization, and output logits remain in plaintext, leaving a non-trivial leakage surface. Scaling to full-model FHE is bounded by the TLU-precision ceiling of `concrete-ml`, ciphertext expansion, and the cost of programmable bootstrapping for non-polynomial operators, and our present evaluation is confined to a 3B model on CPU hardware.

Future work will therefore extend homomorphic coverage to the components not yet encrypted in this paper. We are developing FHE-compatible realizations of the non-linear activations (via low-degree polynomial approximations), the softmax (via programmable bootstrapping with piecewise lookup tables), layer normalization, and the output logits projection, with the goal of eliminating the residual-stream leakage of the current design and moving towards end-to-end encrypted decoding. In parallel, we plan to investigate hardware-accelerated FHE backends (mainly, accelerated GPU-based bootstrapping) and hybrid deployments combining PQC-LLaMA-3 with differential privacy, output clipping, and trusted execution environments, so as to obtain a comprehensive, post-quantum, full privacy-preserving inference stack for sensitive large language-model applications.

CRediT authorship contribution statement

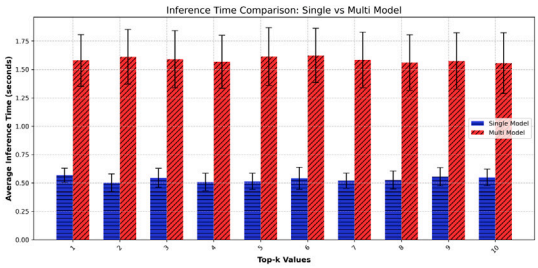
Anes Abdennebi: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Nadjia Kara:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition. **Lahlou Laaziz:** Writing – review & editing, Project administration, Investigation.

Declaration of competing interest

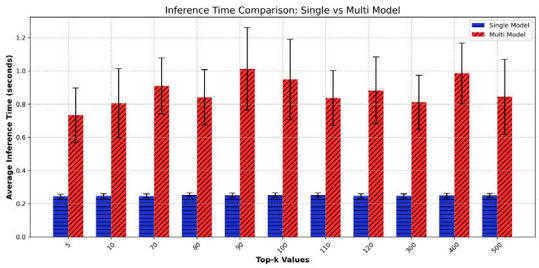
The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Dr. Nadjia Kara reports financial support was provided by Department of National Defence of Canada. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

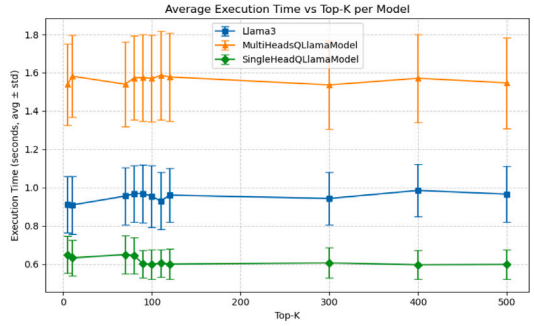
This work was supported by the Department of National Defence of Canada under the Mobilizing Insights in Defence and Security (MINDS - Targeted Engagement Grant) program, Grant No 24-2-62.



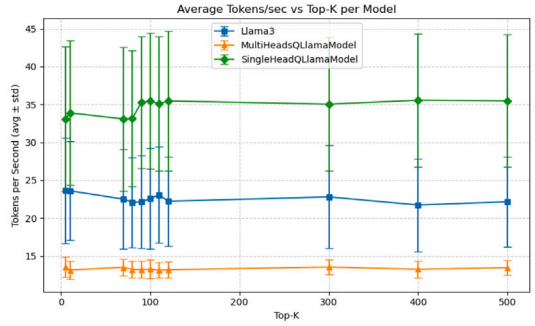
(a) Inference times on Machine `M1` under the FHE mode.



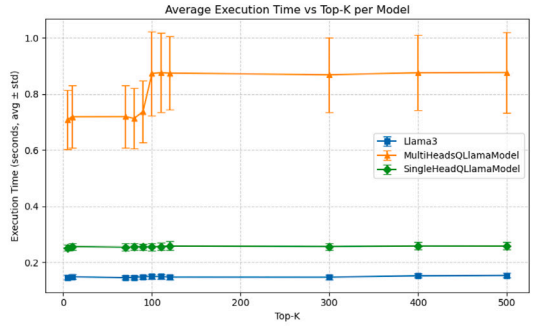
(b) Inference times on Machine `M3` under the FHE mode.



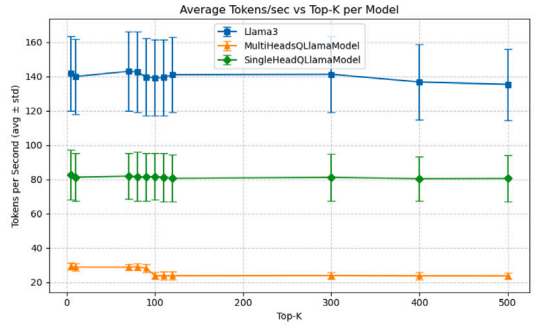
(c) Average text generation times across models on `M1`.



(d) Average tokens generated per second on `M1`.



(e) Average text generation times across models on `M3`.



(f) Average tokens generated per second on `M3`.

Fig. 6. Performance comparison of the PQC-LLaMA-3 and the plain LLaMA-3 models across multiple machines and configurations. Sub Figs. 6(a) and 6(b) show inference time comparisons on different machines, while [6(c), 6(d)] and [6(e), 6(f)] pairs present the averaged performance metrics [`inference times`, `tokens per second`] across models.

Appendix A. Inference times & tokens per second

In this appendix, we focus on the best and the worst performing machines `M1` and `M3`, respectively, from the described evaluation setup in 4. The inference times of the encrypted models `SingleHeadQLlamaModel` and `MultiHeadsQLlamaModel` on both machines `M1` and `M3` (Figs. 6(a) and 6(b)) show the performance lead of the single head encrypted model over the multi-heads one. This behaviour is expected since the computational overhead is greater in the `MultiHeadsQLlamaModel` incurring many more ciphertext multiplications and relinearization (bootstrapping) operations to regularize the noise levels. Moreover, models' uniformity or fluctuation in the execution times is linked to the main (DRAM) and cache memory accesses and their read-write patterns, which depends on their size and the growing memory requirements by the models.

Regarding the Tokens/Sec rates illustrated in Figs. 6(d) and 6(f), the overall observation and analysis show a link between the tokens/sec rates and the processor's compute power and the memory cache's capacity and architecture. The performance comparison in Fig. 6(d) shows a clear differentiation in computational efficiency across the three models. The `SingleHeadQLlamaModel` achieves the highest throughput, averaging ≈ 35-37 tokens/s, outperforming the plain LLaMA-3 model, which sustains ≈ 22-24 tokens/s, and significantly exceeding the `MultiHeadsQLlamaModel`, which achieves ≈ 12-13 tokens/s. This improvement represents an approximate 1.5× increase over the plain model and nearly 3× over the multi-head version, highlighting the efficiency of the single-head FHE-quantized inference configuration.

It is important to note that the above comparison between `SingleHeadQLlamaModel` and plain LLaMA-3 is structurally asymmetric, the plain model runs in FP16/FP32 across all 32 layers, while the FHE variant processes one FHE-encrypted layer and 31 plaintext layers with 2-bit quantization. The observed token-rate advantage of the single-head model on `M1` is best attributed to two factors: (i) reduced memory footprint due to 2-bit quantization of the single encrypted layer, and (ii) the slight L1 cache size advantages as discussed in Appendix B. To provide a fair three-way comparison, Section 4.8 evaluates: (a) plain FP16 LLaMA-3, (b) the same multi-head architecture with quantization but FHE disabled, and (c) the same architecture with quantization and FHE enabled. This three-way design isolates the performance impact of quantization from the impact of FHE encryption.

On the other hand, the models performances are slightly different, mainly the switch between SingleHeadQLlamaModel and LLaMA-3 in the order of performance superiority. The general behaviour for machine M3 exhibits an overall higher tokens/sec rates compared to M1, due to the latter's higher CPU frequency (3.2 GHz compared to 2.65 GHz for base frequency, and 6 GHz compared to 3 GHz for turbo clock). The plain LLaMA-3 model achieves 142 tokens/sec which is 1.7× higher than SingleHeadQLlamaModel (81 token/sec) and 4.7× higher than MultiHeadsQLlamaModel (30 tokens/sec).

This behaviour described in Figs. 6(c),6(e), 6(d),6(f) taken on machines M1 and M3, suggests that the reduction in attention head dimensionality and quantized arithmetic under FHE can lead to faster effective throughput, particularly when cache locality and memory access patterns are optimized. Conversely, the multi-head FHE model exhibits a compounded latency due to increased ciphertext management and parallel homomorphic operations across multiple heads, which increase synchronization and memory overhead. Furthermore, the cache memory architecture in both machines hints an important potential cause for the superiority of SingleHeadQLlamaModel over Llama3 shown in Fig. 6(d). The L1 cache size in M1 is slightly higher than M3's which might be a decisive factor when ciphertexts, activations, and gradients are moved to the cache. The larger size of the cache's first level may accommodate ciphertext matrices for the single head model (in M1) while it might be also moved to the next cache level if the $L1$'s is narrower (as in M3). This memory capacity difference might render the single head encrypted model more efficient than the plain one, only due to the cache memory's capacity.

Appendix B. Cache memory readings

See Fig. 8.

B.1. Cache behaviour on M1

The memory access patterns on the machine M1 is different than M3, the general access pattern shows general steady cache accesses with more dependency on the L1 and L2 caches and less L3/LCC accesses by all PQC-LLaMA-3 models (single and multi). More granular, the MultiHeadsQLlamaModel is more efficient than the SingleHeadQLlamaModel regarding the L1 and L2 cache memory, reaching the following approximate rates; 13% for 15% (Fig. 7(a)), 0.36% for 0.385% (Fig. 7(c)), and 21.6% for 22.2% (Fig. 7(e)), respectively, when comparing these two models. However, this behaviour is reversed while inspecting the L3 cache miss rates, the MultiHeadsQLlamaModel slightly degrades in its performance reaching ≈19% (Fig. 7(g)).

Taken together, these measurements paint a coherent picture of the memory hierarchy behaviour for FHE-quantized inference with *KV-caching*:

- Strong L1 locality (data and instructions). The very low L1d and L1i miss rates imply that the hot inner loops (FHE kernels, attention scoring, and per-token postprocessing) largely operate on data that is quickly reused and remains resident in L1. This outcome is plausible in a KV-cached inference pipeline where most per-token operations reuse small vectors and matrix fragments that are repeatedly accessed during attention computations.
- L2 as the pressure point. The elevated L2 miss rates point to the L2 cache (24 MiB) being the principal capacity/associativity boundary for the workload. Two complementary mechanisms likely explain this:
    1. Working-set size and streaming patterns. FHE-quantized tensors (ciphertext blocks, quantized weight shards) and per-token state can be large. Although the hot inner working set fits in L1, the aggregate data for many tokens and multi-head attention (multiple heads producing/consuming KV pairs) can exceed L2 residency. This leads to L1 misses that find L2 only intermittently.
    2. Parallelism and interleaved streams (multi-head). The multi-head model can create more concurrent memory streams (per-head buffers, parallel attention kernels). Even when total LLC residency is sufficient, the distribution of those streams can increase L2 conflict misses (associativity and replacement interactions) and thus raise L2 miss fractions.
- LLC largely mitigates DRAM traffic. The large 256 MiB LLC provides a high-capacity region for sharing and reusing KV data across tokens and heads. The moderate LLC miss rates indicate that although a non-negligible fraction of accesses escalates beyond L3, a majority of L2 misses are resolved in LLC. This behaviour explains the modest observed variability of end-to-end latency across top-k: as long as KV caches and frequently reused quantized arrays remain resident in L3, increasing top-k does not substantially increase DRAM traffic or invocation latency.

B.2. Cache behaviour on M3

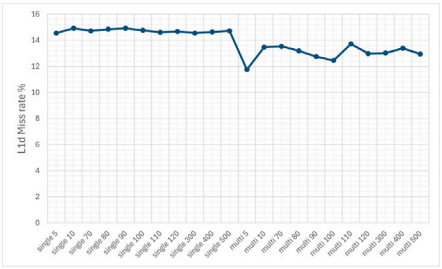
The Figs. 7(b), 7(d), 7(f), 7(h) show the recorded cache memory miss rates in all its levels, L1, L2, and L3/LLC. The general observation for both single head and multi heads models is cache memory dependency and usage efficiency on all levels (except for L2 as a special case), suggesting different spatial and temporal locality behaviours influenced by the encryption overhead and the model's internal parallelism.

The L1d cache's miss rate is extremely low across all top-k values for each model ($\approx 3.5 \times 10^{-12}$), showcasing that the models weight tensors and some activation blocks reside in the first cache memory's level. Nevertheless, there are some micro observations to be made. As shown in Fig. 7(b), the highest miss rates belong to single 5 and multi 5 trials, which is due to the cache warmup condition. The cache needs to be populated before start storing and fetching data for the running program, in our scenario, the L1d needs to be populated with the model's weights so it would retrieve it faster for next calls. Furthermore, the MultiHeadsQLlamaModel has a slightly better miss rate than the SingleHeadQLlamaModel, which can be noticed at the data point starting from multi 10 (we excluded multi 5 due to the warmup step).

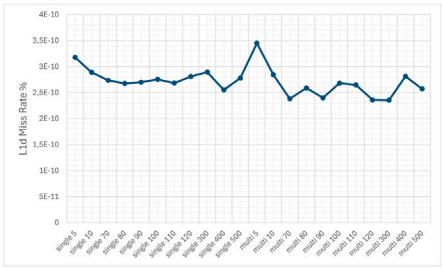
For the L1 instruction cache (L1i), the single-head model shows moderate variation across top-k values, particularly a noticeable increase in miss rate from single 70 to single 100. This is attributed to branch and kernel-switching overhead caused by token-dependent control flow during longer-context inference. The multi-head model maintains much lower L1i miss rates ($\leq 0.1\%$), indicating better instruction-level locality. This improvement likely arises from instruction reuse across parallel attention heads, where identical kernel sequences are reused, reducing instruction fetch misses.

The situation is different with the second level of the cache L2 (Fig. 7(f)). For both models, the miss rate is relatively high (when compared with the other levels), with higher rates for the multi heads model (with averages around 21% for the single-head and 36% for the multi-head model). It appears that the models are less dependent on L2 with more emphasis on the other cache levels. For FHE workloads, such a behaviour may be linked to the intermediate ciphertext buffers, which are relatively large and often would not fit the L2 cache due to frequent eviction when the encrypted attention blocks exceed the L2 capacity. The multi-head model, as noticed in Fig. 7(f) starting from multi 5, arouse this effect because parallel encrypted heads introduce multiple concurrent and large memory streams, increasing L2 contention and decreasing its effective reuse distance.

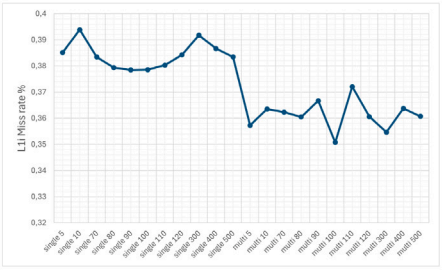
The cache access patterns for L1i (the first cache level for instructions) and LLC (or L3 cache) are quite similar (see Figs. 7(d) and 7(h)). The single head model shows a fluctuating cache access but still efficient (0.15%–0.8%, range including both cache levels L1i and LLC). Moreover, it experiences degradation in the cache access



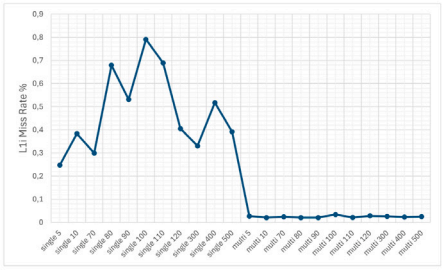
(a) The L1d cache miss rate on the `M1` machine



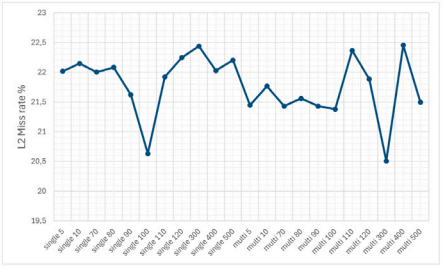
(b) The L1d cache miss rate on the `M3` machine



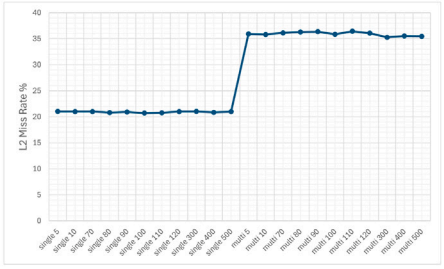
(c) The L1i cache miss rate on the `M1` machine



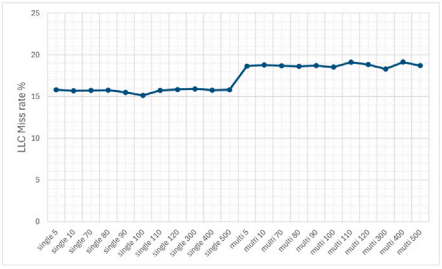
(d) The L1i cache miss rate on the `M3` machine



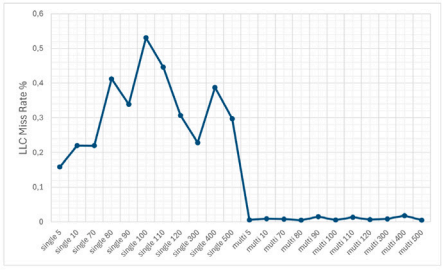
(e) The L2 cache miss rate on the `M1` machine



(f) The L2 cache miss rate on the `M3` machine



(g) The LLC miss rate on the `M1` machine

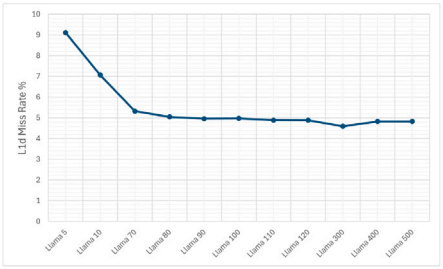


(h) The Last Level Cache (LLC) miss rate on the `M3` machine

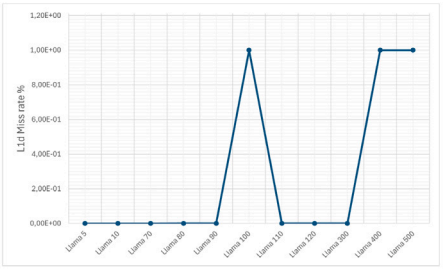
Fig. 7. The cache memory's miss rate for both encrypted `SingleHeadQLlamaModel` and `MultiHeadsQLlamaModel` models.

when moving from short context to a longer context text generation (from 10 to 500 new tokens) on both cache levels, `L1i` and LLC. This can be linked to the sudden change from short to longer new token sequences where the in-between tokens need to be computed first (they were not computed before) then stored in the cache before fetching them. This cache access behaviour would be different if intermediate trials were present between `single` 10 and `single` 70. On the other hand, the multi heads model reaches a perfect LLC hit rate while performing steadily across all top-k values, reflecting effective sharing of precomputed ciphertexts and quantized tensors across the attention heads. The steady LLC hit rate also helps explain why the average inference times remain nearly constant across all top-k values, once the model's working set fits within the LLC, additional tokens do not significantly increase cache misses or latency.

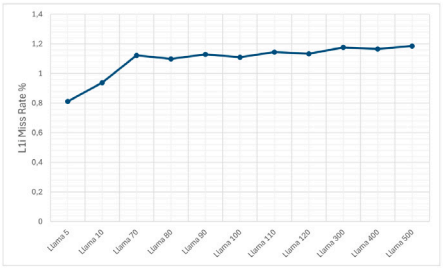
Overall, the results indicate that both models exhibit strong `L1` and LLC locality, but diverge at the L2 level due to architectural and the required memory budget imposed by the integration of FHE primitives and functions within the model. The `MultiHeadsQLlamaModel` demonstrates higher L2 miss rates yet achieves superior LLC efficiency, implying that data sharing at the last-level cache compensates for mid-level cache inefficiency. Consequently, the similar inference times observed across top-k values are justified by the stable `L1d` and LLC hit behaviour, which mitigates the latency impact of variable top-k decoding workloads.



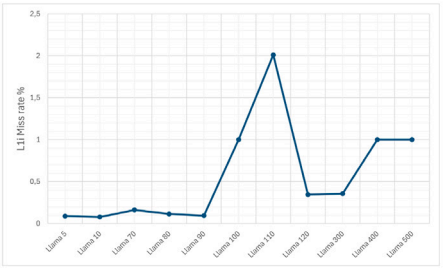
(a) The L1d cache miss rate on the M1 machine for the plain LLaMA-3 model.



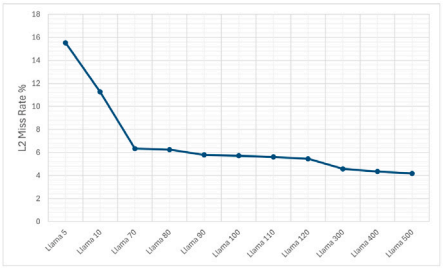
(b) The L1d cache miss rate on the M3 machine for the plain LLaMA-3 model



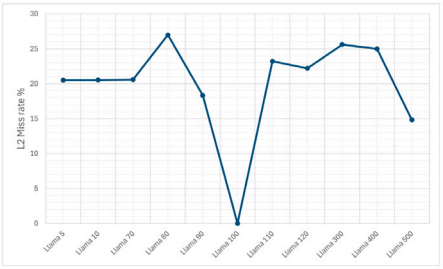
(c) The L1i cache miss rate on the M1 machine for the plain LLaMA-3 model.



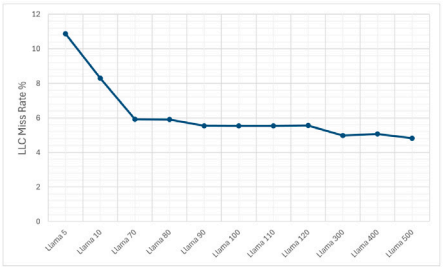
(d) The L1i cache miss rate on the M3 machine for the plain LLaMA-3 model



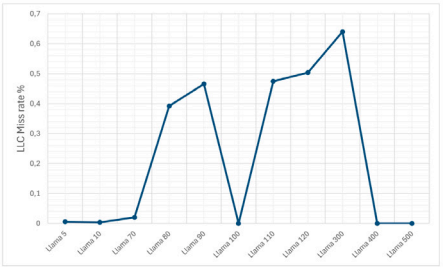
(e) The L2 cache miss rate on the M1 machine for the plain LLaMA-3 model



(f) The L2 cache miss rate on the M3 machine for the plain LLaMA-3 model



(g) The LLC cache miss rate on the M1 machine for the plain LLaMA-3 model



(h) The LLC cache miss rate on the M3 machine for the plain LLaMA-3 model

Fig. 8. The cache memory's miss rate for the plain unencrypted LLaMA-3 model.

B.3. Cache behaviour with LLaMA-3

Fig. 8 presents the cache miss rates for the plain (unencrypted) LLaMA-3 model across different top-k configurations, alongside the previously measured Fully Homomorphic Encryption quantized single-head and multi-head variants. The plain model demonstrates a consistently low miss rate at all cache levels, reflecting a more efficient memory access pattern and reduced cache pressure. Specifically, the L1d data miss rate for the plain model decreases from 9.1% at top-k = 5 to approximately 4.8% beyond top-k = 100 (Fig. 8(a)), representing a 2-3× improvement compared to the encrypted single-head (≈14%–15%) and multi-head (≈12%–13%) counterparts on M1 (Fig. 7(a)). This reduction suggests that the unencrypted model exhibits better temporal and spatial locality in weight and activation reuse, allowing more effective exploitation of the 1.5 MiB L1d data cache.

A similar trend is observed at the L2 and LLC levels, where the plain model's miss rates fall within 4%–15% and 5%–11%, respectively, (see Figs. 8(e) and 8(g)) significantly lower than the encrypted models (20%–22% for L2 and 15%–19% for L3) on machine M1 (Figs. 7(e) and 7(g)). The sharper decline in L2 and L3 miss rates for higher top-k values indicates that larger decoding workloads favour cache reuse under plaintext arithmetic, whereas encrypted execution under FHE

incurs additional tensor expansion and ciphertext-related overhead, which leads to less predictable access patterns and more frequent evictions across all cache levels.

Interestingly in Figs. 7(c) and 8(c), the L1i instruction cache miss rate (≈0.8%–1.2%) for the plain model is slightly higher than in the FHE cases (≈0.36%–0.38%), which can be attributed to the higher instruction-level diversity and deeper execution path of the native LLaMA-3 model. The FHE quantized models, in contrast, exhibit a more uniform execution flow dominated by homomorphic arithmetic kernels with repetitive instruction sequences that fit efficiently within the L1 instruction cache.

The cache access patterns for the plain model on machine M3 (Figs. 8(b), 8(d), 8(f), and 8(h)) have a near binary access mode, either a full cache hit or miss. For instance, when there is a 100% cache miss for Llama 100 settings on the first level cache L1d, it is answered by an access from the third level as shown in Figs. 8(b), 8(f), and 8(h) having a 100% hit rate. The same observation for instances Llama 400 and Llama 500. Furthermore, it seems that the plain model depends less on the second cache level L2 where the miss rates are between 0% and 27%. The cache memory performance for the plain model remains superior to the encrypted models (single and multi heads) on machine M3, however, the difference in performance is too narrow for all cache levels. It is also important to note that the performance of memory access on M3 is more compelling than than its counterpart M1.

Overall, These observations demonstrates an efficient use of the cache memory by the plain model and distinguishes it from the other behaviours of the encrypted models, which can be linked to the extra memory and computation overhead imposed by the additional ciphertexts matrices and homomorphic operations in the attention layers.

These results also indicate that while FHE quantization provides cryptographic security and data privacy guarantees, it introduce some cache inefficiency due to ciphertext expansion, larger memory footprints, and reduced data reuse, which is an expected logic. The plain LLaMA-3 model, benefiting from compact data representation and regular access patterns, maintains high cache locality across all hierarchy levels. These findings highlight the memory access trade-off inherent to privacy-preserving inference, where encrypted computation leads to measurable degradation in cache efficiency and consequently higher memory latency and energy cost.

Data availability

The data that has been used is confidential.

References

[1] S. Singh, F. Abri, A.S. Namin, Exploiting large language models (llms) through deception techniques and persuasion principles, in: 2023 IEEE International Conference on Big Data, BigData, IEEE, 2023, pp. 2508–2517.

[2] M. Mozes, X. He, B. Kleinberg, L.D. Griffin, Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities, 2023, arXiv preprint arXiv:2308.12833.

[3] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, Y. Zhang, A survey on large language model (llm) security and privacy: The good, the bad, and the ugly, High-Confid. Comput. 4 (2) (2024) 100211.

[4] G. Choquet, A. Aizier, G. Bernollin, Exploiting privacy vulnerabilities in open source llms using maliciously crafted prompts, 2024.

[5] D. Kang, X. Li, I. Stoica, C. Guestrin, M. Zaharia, T. Hashimoto, Exploiting programmatic behavior of llms: Dual-use through standard security attacks, in: 2024 IEEE Security and Privacy Workshops, SPW, IEEE, 2024, pp. 132–143.

[6] A. Zou, Z. Wang, N. Carlini, M. Nasr, J.Z. Kolter, M. Fredrikson, Universal and transferable adversarial attacks on aligned language models, 2023, arXiv preprint arXiv:2307.15043.

[7] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A.F. Cooper, D. Ippolito, C.A. Choquette-Choo, E. Wallace, F. Tramèr, K. Lee, Scalable extraction of training data from (production) language models, 2023, arXiv preprint arXiv:2311.17035.

[8] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz, Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection, in: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, 2023, pp. 79–90.

[9] F. Perez, I. Ribeiro, Ignore previous prompt: Attack techniques for language models, 2022, arXiv preprint arXiv:2211.09527.

[10] M.S. Lisa Cabel, Impacts of artificial intelligence in the workplace — kpmg.com, 2024, https://kpmg.com/ca/en/home/insights/2024/09/impacts-of-artificial-intelligence-in-the-workplace.html. (Accessed 19 January 2026).

[11] H. Ken, et al., Generative AI Security: Theories and Practices, Springer Nature Switzerland, 2024.

[12] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, Y. Liu, Masterkey: Automated jailbreak across multiple large language model chatbots, 2023, arXiv preprint arXiv:2307.08715.

[13] V.K. Nguyen, M.I. Husain, Penetration testing of agentic AI: A comparative security analysis across models and frameworks, 2025, arXiv preprint arXiv:2512.14860.

[14] P. Chao, A. Robey, E. Dobriban, H. Hassani, G.J. Pappas, E. Wong, Jailbreaking black box large language models in twenty queries, in: 2025 IEEE Conference on Secure and Trustworthy Machine Learning, SaTML, IEEE, 2025, pp. 23–42.

[15] S. Yi, Y. Liu, Z. Sun, T. Cong, X. He, J. Song, K. Xu, Q. Li, Jailbreak attacks and defenses against large language models: A survey, 2024, arXiv preprint arXiv:2407.04295.

[16] B. Peng, K. Chen, Q. Niu, Z. Bi, M. Liu, P. Feng, T. Wang, L.K. Yan, Y. Wen, Y. Zhang, et al., Jailbreaking and mitigation of vulnerabilities in large language models, 2024, arXiv preprint arXiv:2410.15236.

[17] L. Jiang, K. Rao, S. Han, A. Ettinger, F. Brahman, S. Kumar, N. Mireshghallah, X. Lu, M. Sap, Y. Choi, et al., Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models, Adv. Neural Inf. Process. Syst. 37 (2024) 47094–47165.

[18] J. Chen, K. Li, S.Y. Philip, Privacy-preserving deep learning model for decentralized VANETs using fully homomorphic encryption and blockchain, IEEE Trans. Intell. Transp. Syst. 23 (8) (2021) 11633–11642.

[19] P. Zuo, Y. Hua, L. Liang, X. Xie, X. Hu, Y. Xie, Sealing neural network models in encrypted deep learning accelerators, in: 2021 58th ACM/IEEE Design Automation Conference, DAC, IEEE, 2021, pp. 1255–1260.

[20] Q.-X. Huang, W.L. Yap, M.-Y. Chiu, H.-M. Sun, Privacy-preserving deep learning with learnable image encryption on medical images, IEEE Access 10 (2022) 66345–66355.

[21] B. Balaban, S.S. Magara, C. Yilgor, A. Yucekul, I. Obeid, J. Pizones, F. Kleinstueck, F.J.S. Perez-Grueso, F. Pellise, A. Alanay, et al., Privacy-preserving machine learning (PPML) inference for clinically actionable models, IEEE Access (2025).

[22] K. Cong, D. Das, J. Park, H.V. Pereira, Sortinghat: Efficient private decision tree evaluation via homomorphic encryption and transciphering, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 563–577.

[23] R. Akhavan Mahdavi, H. Ni, D. Linkov, F. Kerschbaum, Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023, pp. 2945–2958.

[24] R. Bost, R.A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data, 2014, Cryptology ePrint Archive.

[25] T. Graepel, K. Lauter, M. Naehrig, ML confidential: Machine learning on encrypted data, in: International Conference on Information Security and Cryptology, Springer, 2012, pp. 1–21.

[26] F.-J. González-Serrano, A. Amor-Martín, J. Casamayón-Antón, Supervised machine learning using encrypted training data, Int. J. Inf. Secur. 17 (4) (2018) 365–377.

[27] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, K. Xu, Machine learning-powered encrypted network traffic analysis: A comprehensive survey, IEEE Commun. Surv. Tutor. 25 (1) (2022) 791–824.

[28] E. Frimpong, K. Nguyen, M. Budzys, T. Khan, A. Michalas, Guardml: Efficient privacy-preserving machine learning services through hybrid homomorphic encryption, in: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, 2024, pp. 953–962.

[29] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, 2024, arXiv preprint arXiv:2407.21783.

[30] C.M. Zama, A privacy-preserving machine learning library using fully homomorphic encryption for data scientists, 2022.

[31] V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 1–23.

[32] V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings, J. ACM 60 (6) (2013) 1–35.

[33] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, 2009, pp. 169–178.

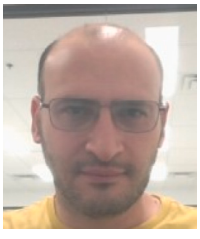
[34] M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 24–43.
[35] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, ACM Trans. Comput. Theory (TOCT) 6 (3) (2014) 1–36.
[36] Z. Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP, in: Annual Cryptology Conference, Springer, 2012, pp. 868–886.
[37] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, 2012, pp. 1219–1234.
[38] C. Gentry, A. Sahai, B. Waters, Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based, in: Annual Cryptology Conference, Springer, 2013, pp. 75–92.
[39] C. Gentry, S. Halevi, Implementing gentry's fully-homomorphic encryption scheme, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2011, pp. 129–148.
[40] J.H. Cheon, J.-S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi, A. Yun, Batch fully homomorphic encryption over the integers, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2013, pp. 315–335.
[41] M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical? in: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, 2011, pp. 113–124.
[42] J.W. Bos, K. Lauter, J. Loftus, M. Naehrig, Improved security for a ring-based fully homomorphic encryption scheme, in: IMA International Conference on Cryptography and Coding, Springer, 2013, pp. 45–64.
[43] C. Gentry, S. Halevi, N.P. Smart, Homomorphic evaluation of the AES circuit, in: Annual Cryptology Conference, Springer, 2012, pp. 850–867.
[44] J.-S. Coron, D. Naccache, M. Tibouchi, Public key compression and modulus switching for fully homomorphic encryption over the integers, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2012, pp. 446–464.
[45] J. Fan, F. Vercauteren, Somewhat practical fully homomorphic encryption, 2012, Cryptology ePrint Archive.
[46] I. Chillotti, N. Gama, M. Georgieva, M. Izabachene, Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2016, pp. 3–33.
[47] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, TFHE: Fast fully homomorphic encryption over the torus, J. Cryptology 33 (1) (2020) 34–91.
[48] L. Bergerat, A. Boudi, Q. Bourgerie, I. Chillotti, D. Ligier, J.-B. Orfila, S. Tap, Parameter optimization and larger precision for (T)FHE, J. Cryptology 36 (3) (2023) 28.
[49] M.R. Albrecht, R. Player, S. Scott, On the concrete hardness of learning with errors, 2015, Journal of Mathematical Cryptology, 9(3):169–203, 2015; lattice-estimator tool available at https://github.com/malb/lattice-estimator, https://lwe-estimator.readthedocs.io/en/latest/.
[50] E. Alkim, L. Ducas, T. Pöppelmann, P. Schwabe, Post-quantum key exchange — A new hope, in: 25th USENIX Security Symposium, USENIX Security 16, 2016, pp. 327–343.
[51] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, V. Vaikuntanathan, Homomorphic Encryption Security Standard, Technical Report, HomomorphicEncryption.org, Toronto, Canada, 2018.
[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[53] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, S. Sanghai, Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023, arXiv preprint arXiv:2305.13245.
[54] I. Beltagy, M.E. Peters, A. Cohan, Longformer: The long-document transformer, 2020, arXiv preprint arXiv:2004.05150.
[55] T. Zhang, V. Kishore, F. Wu, K.Q. Weinberger, Y. Artzi, BERTscore: Evaluating text generation with BERT, in: 8th International Conference on Learning Representations, ICLR 2020, OpenReview.net, Addis Ababa, Ethiopia, 2020, URL https://openreview.net/forum?id=SkeHuCVFDr.
[56] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, BLEU: A method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2002, pp. 311–318, http://dx.doi.org/10.3115/1073083.1073135, URL https://aclanthology.org/P02-1040.
[57] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81, URL https://aclanthology.org/W04-1013.
[58] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S.R. Bowman, GLUE: A multi-task benchmark and analysis platform for natural language understanding, in: International Conference on Learning Representations, 2019, URL https://openreview.net/forum?id=rJ4km2R5t7.
[59] T. Dettmers, M. Lewis, Y. Belkada, L. Zettlemoyer, LLM.int8(): 8-bit matrix multiplication for transformers at scale, NeurIPS, in: Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 30318–30332.
[60] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, S. Han, SmoothQuant: Accurate and efficient post-training quantization for large language models, in: Proceedings of the 40th International Conference on Machine Learning, ICML, 2023, pp. 38087–38099.
[61] C. Song, A. Raghunathan, Information leakage in embedding models, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS, 2020, pp. 377–390.
[62] J.X. Morris, V. Kuleshov, V. Shmatikov, A.M. Rush, Text embeddings reveal (almost) as much as text, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2023, pp. 12448–12460.
[63] H. Li, M. Xu, Y. Song, Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence, in: Findings of the Association for Computational Linguistics: ACL 2023, 2023, pp. 14022–14040.
[64] X. Pan, M. Zhang, S. Ji, M. Yang, Privacy risks of general-purpose language models, in: 2020 IEEE Symposium on Security and Privacy, S&P, IEEE, 2020, pp. 1314–1331.
[65] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, T. Zhang, IRON: Private inference on transformers, NeurIPS, in: Advances in Neural Information Processing Systems, vol. 35, 2022.
[66] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, R. Sharma, CrypTFlow2: Practical 2-party secure inference, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS, 2020, pp. 325–342.
[67] D. Li, R. Shao, H. Wang, H. Guo, E.P. Xing, H. Zhang, MPCFormer: Fast, performant and private transformer inference with MPC, in: International Conference on Learning Representations, ICLR, 2023.
[68] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, F. Wei, THE-X: Privacy-preserving transformer inference with homomorphic encryption, in: Findings of the Association for Computational Linguistics: ACL 2022, 2022, pp. 3510–3520.
[69] W.-j. Lu, Z. Huang, Z. Gu, J. Li, J. Liu, et al., BumbleBee: Secure two-party inference framework for large transformers, in: Network and Distributed System Security Symposium, NDSS, 2025.
[70] Q. Pang, J. Zhu, H. Møller, W. Zheng, C. Cremers, Bolt: Privacy-preserving, accurate and efficient inference for transformers, in: IEEE Symposium on Security and Privacy, S&P, 2024.
[71] W. Li, et al., Traceability and identity protection in smart agricultural IoT system framework based on blockchains, IEEE Trans. Dependable Secur. Comput. 22 (5) (2025) 5335–5351.



Anes Abdennebi Ph.D. Researcher @ École de Technologie Supérieure (ÉTS) Anes Abdennebi is a Ph.D. researcher specialized in applied ML/DL/AI and GenAI models for cybersecurity applications at the host and network levels. With backgrounds in high performance computing, machine learning, cryptography, and GenAI models training, a Master's degree in Computer Engineering from Izmir Institute of Technology, a Master's degree in Computer Science & Engineering from SabancıUniversity he has a strong foundation in both theoretical and practical aspects of employing AI in cybersecurity defensive systems.



Dr. Nadjia Kara Professor @ École de Technologie Supérieure (ÉTS) Dr. Nadjia Kara is a Professor at ÉTS Montreal and leads research in cloud computing, network security, and artificial intelligence. Her work focuses on developing secure and intelligent systems for next-generation networks. She has extensive experience in organizing international workshops and conferences in the field of cybersecurity.



Dr. Lahlou Laaziz Research Associate @ École de Technologie Supérieure (ÉTS) Dr. Laaziz Lahlou is a Professor at ÉTS Montreal specializing in cybersecurity and artificial intelligence. His research interests include machine learning applications in security, network defense systems, and AI-driven threat detection. He collaborates with industry partners to develop innovative security solutions.