Privacy-Preserving State Estimation: An Encrypted Extended Kalman Filter Using CKKS Homomorphic Encryption

MICHAEL SOTULA MASANGU[1], MOANDA NDEKO MOSENGO C.M.[1],
WITESYAVWIRWA VIANNEY KAMBALE[3], KYANDOGHERE KYAMAKYA[1,2]
[1]Faculté Polytechnique, Université de Kinshasa (UNIKIN),
Kinshasa,
DEMOCRATIC REPUBLIC OF THE CONGO

[2]Institute for Smart Systems Technologies,
Universität Klagenfurt,
AUSTRIA

[3]Faculty of Information and Communication Technology,
Tshwane University of Technology,
Pretoria,
SOUTH AFRICA

Abstract: Fully Homomorphic Encryption (FHE) enables mathematical operations directly on encrypted data without decryption. Any operation that a polynomial can approximate can, in principle, be executed under an FHE scheme. To protect cyber-physical systems from eavesdropping on sensitive measurements, we integrate CKKS, an FHE scheme for encrypted real and complex arithmetic, into a state estimator. The estimator is an Extended Kalman Filter (EKF) that fuses GPS and Inertial Measurement Unit (IMU) data to estimate vehicle position, velocity, linear acceleration, yaw angle, and turn rate. We implement CKKS using the Microsoft SEAL library, which supports only a limited number of homomorphic arithmetic operations, creating major challenges for EKF steps such as matrix inversion. We address these constraints with operation-efficient approximations and structured compromises. Frobenius norm analysis shows that the encrypted EKF preserves the precision of the plaintext EKF while reducing data exposure, at the cost of increased latency.

Key-Words: Cyber-physical systems security, Extended Kalman filter, Fully homomorphic encryption, Privacy-preserving computation, Secure state estimation, CKKS scheme

Received: April 11, 2025. Revised: September 7, 2025. Accepted: October 5, 2025. Published: April 21, 2026.

1 Introduction

1.1 Background and Motivation

A Cyber-physical system can effectively integrate cyber and physical components using modern sensors, computing, and network technologies, [1]. Smart vehicles, especially electric ones, are characterized by a significant level of integration of large sensor suites, modern communication and information technologies, as well as advanced computing capabilities in a way that some of them satisfy the requirements to be considered cyber-physical systems or at least to be composed of cyber-physical systems.

These vehicles have introduced new fields of research associated with their new potential capabilities, such as collision avoidance, adaptive cruise control, or even various degrees of self-driving, ranging from simple assistance for relatively basic tasks such as parking to fully autonomous driving. It is important to mention here that these advancements come with some drawbacks, especially when it comes to the security of the users in general, as they delegate some of the tasks that were solely controlled by them and for which they were responsible, thus exposing them to the consequences of cyberattacks such as Denial-of-Service (DoS) attacks or false injection attacks and to the privacy of the users data in particular, which can be subjected to eavesdropping attacks, [2].

In this context, there is a need for solutions that mitigate the security risks associated with the integration of ICT and the reliance on local or distant computing capabilities to control vehicles. This article focuses on the privacy issue, which is particularly important because it is as dangerous as promising. Indeed, from a user's perspective, the dangers of having sensitive information that malicious actors can use if rigorous mechanisms of protection are not put in place coexist with the opportunities in the privacy-preserving machine learning, which are available if security is guaranteed. These opportunities include the possibility of developing services based on privacy-preserving machine learning with smart vehicles providing training data anonymously and securely, eventually acting as elements in a Vehicular Ad Hoc Network

(VANET). These services include promising technologies such as privacy-preserving resource management for more efficient use of the vehicle resources, privacy-preserving crash avoidance, or privacy-preserving trajectory optimization, [2]. Thus, we propose to provide privacy to a Kalman Filter-based state estimator applied to the process made by a smart vehicle in motion, modeled by the Constant Turn Rate and Acceleration Model (CTRA). The filter is very powerful in several aspects because it supports estimations of past, present, and even future states, and it can do so even in the presence of unknowns about the precise nature of the modeled process or the measurement instruments involved in the process, [3].

However, prior to any computation, there is an imperative of decryption, which removes the primary security measure against eavesdropping attacks, [1], in order to make data available for the task; there is a period of total exposure of these sensitive input data and output data to that specific type of attacks. Thus, there is a need to reduce the exposure time associated with the state estimation computations in the context of a smart vehicle. Homomorphic encryption, which allows for computations to be performed directly on encrypted data, was long held back from this conversation because of practical limitations. It is gaining traction as a tool to bypass the need for decryption or, at least, as a tool to significantly reduce the duration of the exposure of sensitive data to eventual malicious actors who would try to access it during computations related to state estimation.

However, challenges remain in order to get to such a solution, as even state-of-the-art homomorphic encryption schemes implementations available, such as the implementation of the CKKS Scheme by the SEAL Library, have yet to fully take on their practical limitations, in particular when it comes to the high running times and the restrictions associated with the amount of operations you can perform homomorphically without having accuracy issues.

1.2 Problem Statement and Research Questions

In cyber-physical systems (CPS), state estimation techniques such as the Extended Kalman Filter (EKF) are widely used for real-time navigation and control. However, these systems often deal with sensitive data, which is vulnerable to eavesdropping attacks when transmitted or processed in an unprotected form. Fully Homomorphic Encryption (FHE) offers a promising solution for processing encrypted data without exposing its contents. However, integrating FHE into complex nonlinear estimators like the EKF poses significant computational and algorithmic challenges, especially when working within the limitations of schemes like CKKS that support only approximate arithmetic and have constrained computational depth. Therefore, we formulate the following research questions (RQs):

- RQ-1: How can the CKKS fully homomorphic encryption scheme be effectively integrated into an Extended Kalman Filter without compromising the accuracy of the state estimation?
- RQ-2: What are the computational limitations imposed by the CKKS scheme (as implemented in Microsoft SEAL), and how do they affect critical EKF operations such as matrix inversion and nonlinear transformations?
- RQ-3:How does the encrypted EKF compare with its plaintext counterpart regarding estimation accuracy, particularly when assessed using Frobenius norm and other relevant metrics?
- RQ-4: What is the trade-off between security (i.e., privacy protection via encryption) and system performance (e.g., latency, computation time) in the proposed implementation?

The remainder of this paper is organized as follows: Section 2 briefly surveys homomorphic encryption integration in CPS, with a particular emphasis on encrypted Kalman Filters. Section 3 presents the mathematical equations and the statistical parameters of the EKF. It also includes an overview of the CKKS scheme and the state model of the moving object. Section 4 describes the implementation and design details of our homomorphic encrypted EKF. Section 5 details the obtained results and the other observed implications of integrating the CKKS homomorphic encryption scheme in the EKF. Section 6 discusses the obtained results, highlighting their strengths and weaknesses. Finally, Section 7 concludes the paper with key insights and future directions.

2 Brief Review of State-of-the-Art Encrypted State Estimation Techniques

The quest to reduce the vulnerability of cyber-physical systems against eavesdropping attacks has led some researchers to investigate the possibility of using homomorphic encryption as a shield to protect their sensitive inputs and outputs, even during state estimation computations, [4]. These studies can be generic for all CPS, such as in [5], or more specialized, like, [6], [7], which focuses on controllers, [8], which focuses on power systems,

or [9], which focuses on Kalman Filters. These controllers are less complex than Kalman filters; they thus require less computational depth when it comes to homomorphic encryption schemes. For example, the Linear Quadratic Gaussian Controller with homomorphic encryption in [7], uses a scheme with a computational depth that limits its reach to degree 2 polynomials. For polynomials of degrees larger than 2, the authors use offline communication and computations to bypass the limit without exposing the results. Such a method on the same homomorphic scheme would not have been efficient in the case of the Kalman filter because of the large number of multiplications involved in the computations.

Recent research endeavors to integrate homomorphic encryption schemes in Kalman filters have particularly drawn our attention. These contributions predominantly employ partially homomorphic encryption (PHE) schemes, such as RSA encryption in the work of [10], or Paillier encryption as utilized by [5], [9]. A standard limitation of these schemes lies in their operation on integer-valued data, which inherently introduces quantization errors that degrade the accuracy of the encrypted state estimator. Although each of these studies proposes strategies to mitigate the impact of such errors, it is noteworthy that only, [9], includes a preliminary analysis of their influence on the overall implementation.

The reliance on PHE obliges the authors to adapt the structure and computations of the estimator to comply with the limited set of operations that can be securely performed under such schemes. To circumvent these structural modifications, the approach presented in [11], advocates using secure multiparty computation (SMC) to perform operations not directly supported by their PHE scheme. Along the same lines, a general approach for privacy-preserving state estimation not restricted to Kalman Filters is provided in [12].

Within this context, our proposed implementation leverages a fully homomorphic encryption (FHE) scheme, the CKKS scheme, which natively supports arithmetic over real numbers. This design choice enables the direct application of the original estimator equations without requiring structural alterations and effectively eliminates the issue of quantization error management. Nevertheless, this advantage comes at the expense of computational efficiency, as FHE schemes are known to be computationally more intensive than their partially homomorphic counterparts. For illustrative purposes, in [9], the encrypted filter achieves a computation time of 1.83738 seconds, compared to 0.497792 seconds for its unencrypted counterpart. This result remains promising in terms of practical applicability.

3 The Extended Kalman Filter (EKF)

The Kalman filter addresses the general problem of estimating the state $x \in \mathbb{R}^n$ of a discrete-time controlled process, governed by the following linear stochastic difference equations 1 and 2:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \tag{1}$$

with a measurement vector $z \in \mathbb{R}^m$ given by:

$$z_k = Hx_k + v_k. \tag{2}$$

The random variables w_k and v_k represent the process noise, which accounts for the imperfections of the mathematical model describing the process dynamics, and the measurement noise, which arises from inherent limitations of the sensing process. These noises are assumed to be mutually independent, zero-mean, white, and normally distributed, as specified in equations 3 and 4:

$$p(w) \sim \mathcal{N}(0, Q), \tag{3}$$

$$p(v) \sim \mathcal{N}(0, R). \tag{4}$$

.

The Kalman filter uses two sets of equations to perform state estimation. Time update equations, which predict the value of the state estimates independently of the measurements using a specific mathematical model that describes the process, and measurement update equations, which use the measurements to adjust the predicted state vector values. We thus define $\hat{x}_{k|k-1} \in \mathbb{R}^n$ as the *a priori* estimate of the state at time step k, and $\hat{x}_{k|k} \in \mathbb{R}^n$ as the *a posteriori* estimate after incorporating the measurements contained in z_k. As displayed in Figure 1, the first-time update which corresponds to the prediction phase uses the initial estimates, and the remaining ones use a posteriori estimates, illustrating the recursive nature of the EKF.

Time update equations (prediction step). The state and covariance are propagated using Eqs. (5)–(6).

$$\hat{x}_{k|k-1} = A\,\hat{x}_{k-1|k-1} + B\,u_{k-1}, \tag{5}$$

$$P_{k|k-1} = A\,P_{k-1|k-1}\,A^T + Q. \tag{6}$$

Here, P is the error covariance matrix, in which the initial value is arbitrarily picked as one of the initial parameters of the filter, with the initial value of the state vector.

Measurement update equations (correction/update step). The gain, corrected state estimate, and corrected covariance are computed via Eqs. (7)–(9).

$$K_k = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R\right)^{-1}, \tag{7}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(z_k - H\hat{x}_{k|k-1}\right), \tag{8}$$

$$P_{k|k} = (I - K_kH)\,P_{k|k-1}. \tag{9}$$

In Eq. (7), K_k is the Kalman gain.

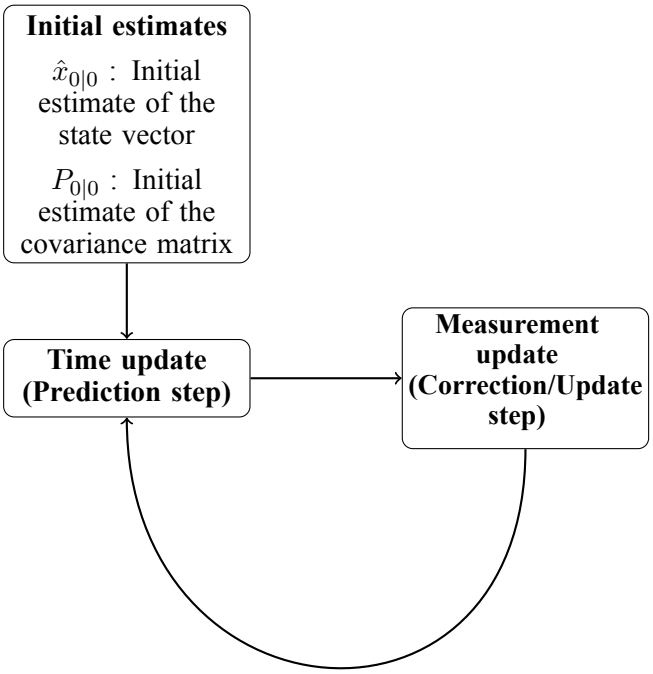

Fig. 1: Recursive operations of Kalman filters, starting from the initial estimates. Source: [3]

However, the procedure is slightly modified when the system is nonlinear. A Kalman filter that is linearized around the current mean and covariance is referred to as an Extended Kalman Filter (EKF). In the EKF, the nonlinear process and measurement models are locally linearized using Jacobian matrices, as defined in Eqs. (10)–(13).

- A is the Jacobian matrix of partial derivatives of f with respect to x:

$$A_{[i,j]} = \left.\frac{\partial f[i]}{\partial x[j]}\right|_{\hat{x}_{k-1|k-1},\, u_{k-1},\, 0}, \tag{10}$$

- W is the Jacobian matrix of partial derivatives of f with respect to w:

$$W_{[i,j]} = \left.\frac{\partial f[i]}{\partial w[j]}\right|_{\hat{x}_{k-1|k-1},\, u_{k-1},\, 0}, \tag{11}$$

- H is the Jacobian matrix of partial derivatives of h with respect to x:

$$H_{[i,j]} = \left.\frac{\partial h[i]}{\partial x[j]}\right|_{\hat{x}_{k-1|k},\, 0}, \tag{12}$$

- V is the Jacobian matrix of partial derivatives of h with respect to v:

$$V_{[i,j]} = \left.\frac{\partial h[i]}{\partial v[j]}\right|_{\hat{x}_{k-1|k},\, 0}. \tag{13}$$

These Jacobian matrices are time-dependent, as the linearization process evolves with the system dynamics. It is also worth noting that analyzing the Taylor remainder when linearizing these functions provides insights into the validity and suitability of using an EKF for nonlinear state estimation. We then obtain new versions of the time update and measurement update equations.

EKF time update equations. The nonlinear state prediction and covariance prediction are performed using Eqs. (14)–(15).

$$\hat{x}_{k|k-1} = f\big(\hat{x}_{k-1|k-1},\, u_{k-1},\, 0\big) \tag{14}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{15}$$

EKF measurement update equations. The Kalman gain, corrected state estimate, and corrected covariance are obtained from Eqs. (16)–(18).

$$K_k = P_{k|k-1} H_k^T \big(H_k P_{k|k-1} H_k^T + R_k\big)^{-1} \tag{16}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\big(z_k - h(\hat{x}_{k|k-1},\, 0)\big) \tag{17}$$

$$P_{k|k} = (I - K_k H_k)\, P_{k|k-1} \tag{18}$$

State model of the moving object. The process under study is the motion of an automobile equipped with a limited set of sensors, including an inertial measurement unit (IMU), a temperature sensor, and a GPS sensor. The objective is to estimate its state vector with high accuracy while minimizing latency, even when the data are encrypted using the CKKS homomorphic encryption scheme. The motion model for the automobile is the CTRA model, which has a state vector with six components. The continuous-time state vector is defined in Eq. (19), and its discrete-time evolution over a sampling period T is given by Eq. (20).

$$\overrightarrow{x}(t) = \begin{pmatrix} x \\ y \\ \psi \\ v \\ a \\ \dot{\psi} \end{pmatrix} \tag{19}$$

The evolution function

$$\overrightarrow{x}(t+T) = \begin{pmatrix} x(t) + \frac{1}{\dot{\psi}^2}[(v\dot{\psi} + a\dot{\psi}T)\sin(\dot{\psi}T+\psi) \\ +a\cos(\dot{\psi}T+\psi) - v\dot{\psi}\sin\psi - a\cos\psi] \\ y(t) + \frac{1}{\dot{\psi}^2}[(-v\dot{\psi} - a\dot{\psi}T)\cos(\dot{\psi}T+\psi) \\ +a\sin(\dot{\psi}T+\psi) + v\dot{\psi}\cos\psi - a\sin\psi] \\ \dot{\psi}T + \psi \\ v + aT \\ a \\ \dot{\psi} \end{pmatrix} = f(\overrightarrow{x}(t)). \tag{20}$$

This evolution function, when linearized at the operating point, yields the Jacobian matrix of the state transition model used in the covariance propagation of Eq. (15). The corresponding Jacobian is given in Eq. (21).

$$J_A = \frac{d(f(\overrightarrow{x}(t)))}{d(\overrightarrow{x}(t))} = \begin{pmatrix} 1 & 0 & a_{13} & a_{14} & a_{15} & a_{16} \\ 0 & 1 & a_{23} & a_{24} & a_{25} & a_{26} \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{21}$$

With:

$$a_{13} = \frac{-\dot{\psi}v\cos(\psi) + a\sin(\psi) - a\sin\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} + \frac{(T\dot{\psi}a - \dot{\psi}v)\cos\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} \tag{22}$$

$$a_{14} = \frac{-\dot{\psi}\sin(\psi) + \dot{\psi}\sin(T\dot{\psi}+\psi)}{\dot{\psi}^2} \tag{23}$$

$$a_{15} = \frac{T\dot{\psi}\sin\left(T\dot{\psi}+\psi\right) - \cos(\psi) + \cos\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} \tag{24}$$

$$a_{16} = \frac{-Ta\sin\left(T\dot{\psi}+\psi\right) + T(T\dot{\psi}a + \dot{\psi}v)\cos\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} + \frac{-v\sin(\psi) + (Ta+v)\sin\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} - \frac{2\left(-\dot{\psi}v\sin(\psi) - a\cos(\psi) + a\cos\left(T\dot{\psi}+\psi\right)\right)}{\dot{\psi}^3} - \frac{2\left((T\dot{\psi}a + \dot{\psi}v)\sin\left(T\dot{\psi}+\psi\right)\right)}{\dot{\psi}^3} \tag{25}$$

$$a_{23} = \frac{-\dot{\psi}v\sin(\psi) - acos(\psi) + a\cos\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} - \frac{(-T\dot{\psi}a - \dot{\psi}v)\sin\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} \tag{26}$$

$$a_{24} = \frac{\dot{\psi}\cos(\psi) - \dot{\psi}\cos(T\dot{\psi}+\psi)}{\dot{\psi}^2} \tag{27}$$

$$a_{25} = \frac{-T\dot{\psi}\cos\left(T\dot{\psi}+\psi\right) - \sin(\psi) + \sin\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} \tag{28}$$

$$a_{26} = \frac{Ta\cos\left(T\dot{\psi}+\psi\right) - T(-T\dot{\psi}a - \dot{\psi}v)\sin\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} + \frac{v\cos(\psi) + (-Ta-v)\cos\left(T\dot{\psi}+\psi\right)}{\dot{\psi}^2} - \frac{2\left(\dot{\psi}v\cos(\psi) - a\sin(\psi) + a\sin\left(T\dot{\psi}+\psi\right)\right)}{\dot{\psi}^3} - \frac{2\left((-T\dot{\psi}a - \dot{\psi}v)\cos\left(T\dot{\psi}+\psi\right)\right)}{\dot{\psi}^3} \tag{29}$$

The observation equation that characterizes the measurement process is given in Eq. (30). This measurement function corresponds to $h(\cdot)$ used in the term of Eq. (17).

$$h(\overrightarrow{x}(t)) = \begin{pmatrix} x \\ y \\ v \\ a \\ \dot{\psi} \end{pmatrix} \tag{30}$$

The measurement dataset comes from the sensor suite; it is available online as referenced in [13]. The Kalman filter proposed in [14], served as a base for the calculations on the unencrypted EKF. The process and measurement noises are modeled with the covariance matrices **Q** and **R** according to [15]. In particular, the state-wise components of **Q** are defined in Eqs. (31)–(36), with the numerical values reported in Table 1.

$$\mathbf{Q} = \begin{pmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\psi^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_a^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{\psi}}^2 \end{pmatrix} \tag{31}$$

$$\sigma_x = \sigma_y = \frac{a_{\max}(\Delta t)^2}{2} \tag{32}$$

$$\sigma_\psi = \Omega_{\max} T \tag{33}$$

$$\sigma_v = a_{\max} T \tag{34}$$

$$\sigma_{\dot{\psi}} = \alpha_{\max} T \tag{35}$$

$$\sigma_a = \frac{1}{2} \tag{36}$$

Table 1: The Extended Kalman filter noise parameters. Source: [14]

Parameter	EKF
σ_x [m]	0.0017
σ_y [m]	0.0017
σ_ψ [rad]	0.002
σ_v [m/s]	0.176
σ_a [m/s^2]	0.5
$\sigma_{\dot{\psi}}$ [rad/s]	0.02

EKF: Extended Kalman Filter.

The measurement noise covariance matrix **R** is defined using the squared standard deviations of the measurement components in Eq. (30). In this work, **R** is set as in Eq. (37).

$$\mathbf{R} = \begin{pmatrix} 36 & 0 & 0 & 0 & 0 \\ 0 & 36 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{37}$$

EKF consistency: NIS. The **Normalized Innovation Squared (NIS)** is a key metric used to assess a state estimator's statistical consistency and unbiasedness. A filter is considered statistically consistent when the distribution of its NIS values passes the chi-squared goodness-of-fit test, as discussed in [16]. In practice, higher statistical consistency reflects a better alignment between the theoretical and empirical performances of the estimator, particularly in its self-assessment via the error covariance matrix **P**. The NIS is computed as in Eq. (38).

$$NIS_k = (\mathbf{Z}_{k|k-1} - \hat{\mathbf{Z}}_k)^T \mathbf{S}_{k|k-1}^{-1} (\mathbf{Z}_{k|k-1} - \hat{\mathbf{Z}}_k) \tag{38}$$

When the ground truth of the state vector is known, two additional metrics can be used: **the Normalized Estimation Error Squared (NEES)** and the **Root Mean Square Error (RMSE)**, which provide insights into estimation accuracy. These are defined as follows: When ground-truth states are available, two additional metrics are commonly used for consistency and accuracy assessment: the **Normalized Estimation Error Squared (NEES)** and the **Root Mean Square Error (RMSE)**. They are given by Eqs. (39)–(40).

$$NEES_k = (\hat{\mathbf{X}}_{k|k} - \mathbf{X}_k^{true})^T \mathbf{P}_{k|k}^{-1} (\hat{\mathbf{X}}_{k|k} - \mathbf{X}_k^{true}) \tag{39}$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=0}^{N} (\hat{\mathbf{X}}_{k|k} - \mathbf{X}_k^{true})^2} \tag{40}$$

Where:

- $\mathbf{Z}_k$ is the measurement vector or the output,
- $\hat{\mathbf{Z}}_{k|k-1} = h(\hat{x}_{k|k-1})$ is the predicted measurement based on the a priori state estimate,
- $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k$ is the innovation covariance matrix, where the innovation ($\nu_k = \mathbf{Z}_{k|k-1} - \hat{\mathbf{Z}}_k$)
- $\mathbf{P}_{k|k}$ is the a posteriori error covariance matrix (a priori when there is $k|k-1$)
- $\mathbf{X}_k^{true}$ is the ground truth of the state vector.

For advanced driver-assistance systems (ADAS), the estimator's positioning error is typically required to remain below $0.3\,\text{m}$, with $0.1\,\text{m}$ often cited as a recommended threshold for reliable operation, [17]. In our study, the true state vectors are not available. Therefore, to select the most accurate motion model prior to the integration of the homomorphic

encryption scheme, we referred to the findings in [18], which indicate that increasing the complexity of motion models does not necessarily yield significant accuracy improvements beyond the Constant Turn Rate and Acceleration (CTRA) model.

In our study, ground-truth state vectors are not available. Therefore, prior to integrating the homomorphic encryption scheme, the model selection relied on the conclusions in [18], which suggest that increasing motion-model complexity does not necessarily provide meaningful accuracy gains beyond the Constant Turn Rate and Acceleration (CTRA) model introduced in Eq. (20).

Table 2: NIS distribution across three different EKF. Source: Created by the authors.

Model	NIS_{max}	NIS_{avg}	NIS_{min}
CTRV	4.5556	0.5408	5.0368 $e^{-}6$
CTRA	120.3729	2.2042	5.1151 $e^{-}6$
CCA	120.2709	2.0285	5.1151 $e^{-}6$

The EKF differ from each other in the motion model used.

Consequently, we evaluated the Constant Turn Rate and Velocity (CTRV), Constant Curvature and Acceleration (CCA), and CTRA models from a statistical consistency perspective using the NIS metric as described in Table 2. Our results go in the same direction as the conclusion of [18], which focuses on accuracy, showing that the CTRA model exhibited slightly better statistical consistency in our specific case. This is illustrated in Figure 2 where you can see that the CCA and CTRA models have a NIS with a better shape compared with the CTRV model in the context of chi-squared goodness-of-fit test. The numerical results are presented in Table 3, where the mean NIS value of 2.4042 for the CTRA model is closer to the lower bound of the respective chi-squared confidence interval compared to 2.0285 and 0.5408 for the two other models. It ultimately means that although all three EKF versions result in a *pessimist filter*, the EKF based on the CTRA model is less pessimistic than the others.

SEAL Library

SEAL is an open-source homomorphic encryption library developed by the Privacy Research Group at Microsoft, [19]. This C++-based library offers a Python binding that enables its use through Python, a tool that we employed in this work to carry out our computations, [13]. It offers a choice between a set of two schemes, namely:

- BFV (Brakerski-Fan-Vercauteren): This scheme only allows integer arithmetic. In its theoretical version, it involves, among other things, efficient packing (SIMD), fast scalar multiplication, fast linear functions, and efficient leveled design. However, it has a slow bootstrapping and is slow when evaluating nonlinear functions.

- CKKS (Cheon-Kim-Kim-Song): This scheme allows real-number arithmetic, which makes it more suitable for applications involving real numbers as parameters. It is leveled because the number of chained multiplications or additions you can perform on one ciphertext without bootstrapping has an upper limit, which depends on the initial parameters set. However, it also has a slow bootstrapping and requires careful manipulation to get a specific level of accuracy because it performs approximate arithmetic.

In the SEAL library, the encryption context is characterized by several parameters, including:

- *Poly Modulus Degree*: Denoted by N, this is a power of two corresponding to the degree of the polynomial in the quotient polynomial ring. Increasing this parameter enables a higher number of computation levels and thus allows for more operations to be performed. This is particularly crucial for multiplicative depth, the maximum number of successive multiplications that can be applied to a ciphertext before the noise level becomes critical.

Table 3: χ^2 goodness-of-fit test results. Source: Created by the authors.

Model	$\frac{bound_{lower}}{N}$	$\frac{bound_{upper}}{N}$	Result
CTRV	3.9468	4.0535	$0.5408 << 3.9468$
CTRA	4.9405	5.0598	$2.2042 < 4.9405$
CCA	4.9405	5.0598	$2.0285 < 4.9405$

N=10800 is the size of the measurement vector. In the result column, the average value of the NIS is compared to the confidence interval's upper and/or lower bound.

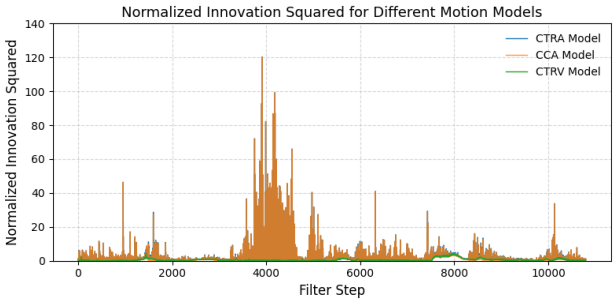

Fig. 2: NIS distribution for three different motion models in an EKF. Source: Created by the authors.

- The vector of prime factors, where each element represents a defined number of bits, is used to configure the *Coefficient Modulus*. This vector includes two types of primes: outer primes (at both ends of the vector), typically chosen equal to achieve symmetry, and inner primes, which must all be identical. The product of these primes defines the full modulus Q.

- The scaling factor, denoted *scale*. Referred to as Δ, this is a power of 2, typically set equal to the value of the inner primes in the coefficient modulus vector. This multiplicative factor is critical in minimizing encryption errors by ensuring many ciphertext representations for the same plaintext value. A larger Δ offers better precision and security at the cost of higher latency.

Finally, the following operations are supported in the CKKS implementation provided by the SEAL library:

- **Add**: Addition and subtraction between ciphertexts.

- **PtMult**: Multiplication between a ciphertext and a plaintext.

- **Mult**: Multiplication between ciphertexts.

- **ModDown**: This can be performed using either ***rescale*** or ***modswitch***.

- **KSKIP**

- **Rotate**: Rotation of slots within a ciphertext.

The theoretical version of the CKKS scheme includes more operations, as explained in [20], where the theoretical framework associated with the library is explained in detail in the second chapter. In particular, bootstrapping is not available. Consequently, it turns this CKKS implementation from a practical standpoint into a leveled homomorphic encryption scheme compared to a classical fully homomorphic encryption scheme.

The main challenges posed by state estimation in the encrypted domain with the SEAL CKKS scheme lie in the subsequent additional calculations required because of the need to compute the Jacobian matrices to obtain the estimates. Thus, we must develop mechanisms to overcome the limitations regarding the depth of the operations we can perform on encrypted data.

4 Design of the Encrypted Extended Kalman Filter

The experimental setup is a Dell Latitude 5440 with 16 GB of RAM and an Intel(R) Core i5-1345U processor. Several operations performed by the unencrypted filter, deemed critical due to the sensitive nature of the data involved, were identified. It was then necessary to determine an appropriate procedure to implement these operations using the tools available: addition (subtraction), multiplication, and rotation (i.e., the rearrangement of data within the polynomials representing the ciphertext). Once the polynomial approximation-based implementation, potentially involving rotations, is finalized, tests are conducted to assess execution time and resource consumption. The latter is evaluated with respect to the parameters defining the encryption context. Thus, in a precision-oriented approach at the expense of latency, we systematically set inner primes to 40 bits and outer primes to 60 bits. The parameter N is selected based on the number of multiplications required by the function to be implemented. Depending on the complexity of the homomorphic function, N is chosen from among the following values:

- 8192
- 16384
- 32768

The various strategies employed to ensure accurate operation and to reduce system latency as much as possible can be summarized in two main elements:

- Ciphertext replication
- Context duplication

Ciphertext Replication This technique replicates a ciphertext as many times as required, ensuring that each use of the ciphertext within a function affects only one copy. For example, given a ciphertext c containing the yaw angle, which is used in the computation of several trigonometric parameters during the Jacobian evaluation (see Equation 21), this method allows us to compute the cosine of the same angle multiple times using different copies. This helps to minimize the cumulative impact associated with the multiplicative depth of the operation. If only one copy of the ciphertext is available and the multiplicative depth is 18, then computing the cosine three times would not be feasible, as it would require at least 21 levels (assuming that only the cosine computation is performed, which is not the case).

This approach allowed us to save multiplication slots at the expense of significantly increased memory consumption since ciphertexts occupy considerably more than their corresponding plaintexts. As a result, RAM usage typically exceeds 90% when executing the encrypted filter script.

Context Duplication This approach aims to reduce execution time, as the filter update and the matrix operations following the prediction and Jacobian computation do not require as many multiplications. A context with larger parameters is suited to the initial, more computationally intensive operations (prediction and Jacobian evaluation) but would be inefficient for the later steps (additional matrix operations and update), leading to resource waste and latency overhead without any corresponding security benefit. However, the drawback of this method is that data becomes vulnerable during transitions between contexts since each context uses its own encryption and evaluation keys.

First Context Parameters: The first context, intended to support high-resource operations, is defined as follows:

- Modulus_degree: $N = 32768$
- Prime factor vector:

$$[60, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 60]$$

, yielding a multiplicative depth of 18

- Scale: $\Delta = 2^{40}$
- Operations performed within this context: (14) and (21).

Second Context Parameters: The second context, intended for lower-resource operations, is defined as follows:

- Modulus_degree: $N = 16384$
- Prime factor vector: $[60, 40, 40, 40, 40, 40, 40, 40, 60]$, corresponding to a multiplicative depth of 7
- Scale: $\Delta = 2^{40}$
- Operations performed within this context: (15), (16), (17) and (18).

To ensure a secure transition between the two contexts, we propose execution within a secure enclave as illustrated in the Figure 3. Specifically, we suggest using Intel's SGX (Software Guard Extensions) technology. SGX is based on another Intel technology known as TXT (Trusted Execution Technology), [21], and provides a trusted computing environment that guarantees confidentiality and integrity. It allows an application to be executed within an enclave-protected container. An application is split into secure and non-secure parts — not based on their actual vulnerability but on the criticality of the associated functions. This framework directs sensitive functions and data into the enclave via a dedicated instruction set to protect these critical computations.

In our case, context-switching operations, which involve decryption and re-encryption, as well as the matrix inversion prior to Kalman Gain computation and the decryption keys of the first context, are considered non-secure and are thus assumed to be transferred into the secure enclave.

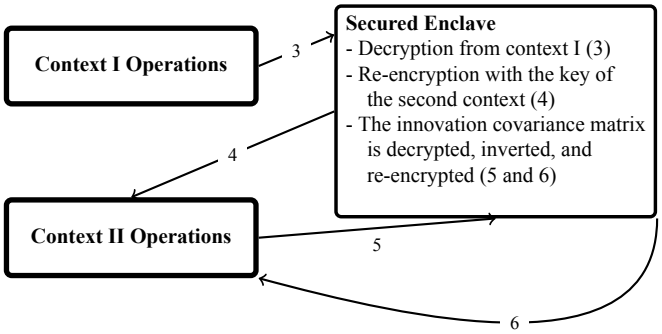

Fig. 3: Interaction Between Context I and II Operations via an External Enclave (with Secure Enclave Usage as a Proposed Enhancement), Steps 1 and 2, preceding Step 3, are preprocessing operations external to the filter itself. Source: Created by the authors.

The critical operations of the EKF, which need to be performed on encrypted data, are the following:

- Addition and subtraction (For scalars and matrices)
- Scalar inversion
- Matrix transposition
- Matrix multiplication
- Sine function
- Cosine function

Homomorphic Addition and Subtraction: Addition and subtraction between two ciphertexts are performed using the library's dedicated function. However, this function, like all others involving two operands, requires the operands' scales and modulus

to be equal. Consequently, we implemented utility functions that take two ciphertexts and align these two parameters before performing the addition or subtraction.

Homomorphic Scalar Multiplication: The case of multiplication between ciphertexts containing scalar values is similar to that of addition and subtraction. Although a predefined function exists for this purpose, the need to align the scale and modulus led us to implement a function that performs this adjustment prior to multiplying ciphertexts containing scalars or before carrying out a mixed multiplication (PtMult) between a ciphertext and an encoded plaintext.

Homomorphic Scalar inversion: Scalar inversion is approximated using the iterative Newton-Raphson method. Its iterative expression is recalled as follows:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{41}$$

where x_{n+1} denotes the $(n+1)$-th term in the approximation of the value of x that satisfies $f(x) = 0$.

By considering $y = \frac{1}{a}$ as the inverse of an arbitrary scalar a, we can define $f(y) = \frac{1}{y} - a$ and substitute it into Equation 41, yielding:

$$y_{n+1} = y_n(2 - ay_n) \tag{42}$$

where a is the Scalar to be inverted.

Depending on the number n of iterations, this approximation requires the equivalent of $2n$ multiplications, which constitutes a significant drawback given our limited *multiplicative budget*. The choice of the initial parameter y_0 also plays a crucial role in the accuracy of the approximation. The results obtained for different values of y_0 indicate local convergence, with the convergence domain depending on the choice of y_0. However, in the case of our filter, the parameter subject to homomorphic inversion lies within the interval $[-0.7769, 0.595]$. Since the range $[-1, 1]$ contains a vast domain of inversion values, it is difficult to cover the whole area with a single choice of y_0; this issue adds to the already high multiplicative cost of the operation.

Consequently, scalar inversion in the filter's prediction phase is performed as a preprocessing step immediately before encryption using the first context.

Homomorphic matrix multiplication: Matrix multiplication is one of the most critical operations to secure in our implementation. Before delving into the method used for secure multiplication, we present the encryption and decryption algorithms for matrices. These two algorithms are essential for understanding the details of the matrix multiplication procedure.

•**Matrix encryption**

Table 4: Matrix Encryption with SEAL CKKS. Source: created by the authors.

Algorithm: Flatten–Encode–Encrypt (for CKKS)

Input: Matrix M; encoder ckks_encoder; encryptor encryptor; scale scale
Initialize empty list flat_list
for each row L in M **do**
    for each element x in L **do**
        Append x to flat_list
    end for
end for
$P \leftarrow$ ckks_encoder.encode(flat_list, scale)
$C \leftarrow$ encryptor.encrypt(P)
return C

A matrix is thus encrypted into a single ciphertext after being flattened, typically using row-wise concatenation as shown in Table 4 The resulting array is then encoded and encrypted. This method enables matrix addition and subtraction using the same functions designed for scalar operations. Therefore, a matrix of size $m \times n$ will occupy $m \times n$ slots in the ciphertext, organized so that the first element of the first row is in slot 0, the first element of the second row is in slot n, etc. with the remaining slots filled with encrypted zeros.

•**Matrix decryption**

Decryption is performed using Table 5, which reconstructs the matrix from the decoded vector of the ciphertext, considering the matrix dimensions to discard irrelevant values.

Table 5: Matrix Decryption with SEAL CKKS. Source: created by the authors.

Algorithm: Decrypt–Decode–Unflatten (matrix reshape)

Require: Ciphertext C; decryptor; ckks_encoder; rows r; cols c
Ensure: Decrypted matrix M
$P \leftarrow$ decryptor.decrypt(C)
flat_list $\leftarrow$ ckks_encoder.decode(P)
$M \leftarrow [\,]$
for $i \leftarrow 0$ **to** $r - 1$ **do**
    $M[i] \leftarrow$ flat_list$[i \cdot c : (i+1) \cdot c]$
end for
return M

The matrix multiplication and transposition algorithms are based on the algorithms used to

encrypt and decrypt matrices. The first one is described by Table 6, presented in a simplified form without the exception-handling logic in the actual implementation. This relatively straightforward multiplication method is based on the exclusive exploitation of the leading slots to compute the dot product (row-column) during matrix multiplication and on the smart use of ciphertext rotations. Once the multiplication is completed, an additional operation is performed in both cases to consolidate the result into a single ciphertext per the matrix encryption scheme described earlier.

It is also worth noting that, despite its apparent simplicity, our homomorphic matrix multiplication yields results that are sufficiently close with an absolute error below 0.01 to those obtained with the matrix multiplication method proposed by [22], as implemented illustratively in the SEAL-Python library, [13]. Moreover, as shown in Table 7,our method is on average faster (1.998 s vs. 7.363 s) when tested on 4×4 matrices, and requires fewer multiplications, two compared to 4 in the library example.

Homomorphic matrix transposition: The matrix transposition operation, described in Table 8, requires only a single multiplication, which is used to reorganize the resulting ciphertext. This operation is similar to homomorphic matrix multiplication, moving a value from the first slot to its appropriate target slot before being added to the final ciphertext. A comparison between our matrix transposition method and the one proposed in the library, following an approach similar to that used for matrix multiplication performances, is presented in Table 9, these results were also obtained using randomly selected 4×4 matrices.

Table 6: Multiplication of two encrypted matrices with SEAL CKKS. Source: created by the authors.

Algorithm: Rotate–Multiply–Accumulate (Packed)

Require: $A \in \mathbb{C}^{n\times m}$, $B \in \mathbb{C}^{m\times k}$; evaluator, galois_keys, relin_keys, mask_e
Ensure: $C \in \mathbb{C}^{n\times k}$ (encrypted)

result_ciphertext $\leftarrow$ [ ]; final_cipher $\leftarrow$ **nil**; $ind \leftarrow 0$
for $i \leftarrow 0$ **to** $n-1$ **do**
  row_results $\leftarrow$ [ ]
  for $j \leftarrow 0$ **to** $k-1$ **do**
    sum_cipher $\leftarrow$ **nil**
    for $l \leftarrow 0$ **to** $m-1$ **do**
      $cell\_a \leftarrow$ evaluator.rotate_vector(A, $i \cdot m + l$, galois_keys)
      $cell\_b \leftarrow$ evaluator.rotate_vector(B, $l \cdot k + j$, galois_keys)
      $tmp \leftarrow$ evaluator.multiply($cell\_a$, $cell\_b$)
      evaluator.relinearize_inplace(tmp, relin_keys)
      evaluator.rescale_to_next_inplace(tmp)
      if sum_cipher **is nil then**
        sum_cipher $\leftarrow tmp$
      else
        evaluator.add_inplace(sum_cipher, tmp)
      end if
    end for
    row_results.push(sum_cipher)
    $temp\_cipher \leftarrow$ Ciphertext(sum_cipher)
    $temp\_cipher \leftarrow$ evaluator.multiply_plain($temp\_cipher$, mask_e)
    load_management("c2p", $temp\_cipher$, relin_keys, evaluator)
    $temp\_cipher \leftarrow$ evaluator.rotate_vector($temp\_cipher$, $-ind$, galois_keys)
    if $ind = 0$ **then**
      final_cipher $\leftarrow temp\_cipher$
    else
      evaluator.add_inplace(final_cipher, $temp\_cipher$)
    end if
    $ind \leftarrow ind + 1$
  end for
  result_ciphertext.push(row_results)
end for
return final_cipher

Table 7: Matrix multiplication performances. Source: Created by the authors.

Matrix multiplication	Execution time [s]	Multiplicative depth
Proposed scheme	1.998	2
Library example	7.363	4

Our version is much faster than the library example when running under the same conditions.

Homomorphic cosine and homomorphic sine: To compute the cosine of an encrypted angle, as with the sine of an encrypted angle, we evaluate a polynomial approximation directly on the ciphertext (which contains the angle) using the Taylor series expansion truncated to six terms. This approach represents a trade-off between the improved accuracy gained by increasing the number of terms and the increased computational cost incurred, particularly due to the higher number of multiplications required. Such an increase in multiplicative depth could render the computation infeasible if it exceeds the allowable limits of the homomorphic encryption context.

A preprocessing step is applied to mitigate the convergence limitations of the truncated Taylor approximation, which may diverge or yield insufficiently accurate results beyond a certain range. This step brings the angles into the interval $[\frac{-10\pi}{8}, \frac{10\pi}{8}]$ within which the sine and cosine values are preserved for use in their respective computations. This normalization ensures that the approximation remains within a window where the truncated Taylor series provides sufficiently precise results. On one hand, the polynomial approximation used for the cosine function is given by:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} \quad (43)$$

On the other hand, the polynomial approximation used for the sine function is given by:

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} \quad (44)$$

The preprocessing function transforms an angle outside the convergence interval $[-10\pi/8, 10\pi/8]$ into a new angle within the convergence zone that preserves the same cosine and sine values. The procedure, described in Table 10, consists of first bringing the angle into the range $[-2\pi, 2\pi]$ then adding or subtracting a value of 2π, depending on whether the intermediate angle (within $[-2\pi, 2\pi]$) is positive or negative. Since adding or subtracting 2π does not alter the value of an angle, the transformation ensures that all trigonometric properties are preserved while mapping the angle into the convergence interval. It is described by the algorithm below:

Table 8: Transposition of a matrix encrypted with SEAL (CKKS) using the proposed matrix encoding. Source: created by the authors.

Algorithm: Rotate-Transpose (Packed)

Require: cipher_m (encrypted $n \times m$ matrix)
rows $= n$, cols $= m$; evaluator, galois_keys, relin_keys, mask_e

Ensure: encrypted transpose $m \times n$

final_cipher $\leftarrow$ **nil**
for $j \leftarrow 0$ **to** $n-1$ **do**
  $ind \leftarrow j$
  for $i \leftarrow 0$ **to** $m-1$ **do** ▷ place $A[i][j]$ into position $A[j][i]$ via slot shifts
    $trans\_cipher \leftarrow$ evaluator.rotate_vector(cipher_m, $i + j \cdot m$, galois_keys)
    $temp\_cipher \leftarrow$ Ciphertext($trans\_cipher$) ▷ mask and realign the encrypted value
    $temp\_cipher \leftarrow$ evaluator.multiply_plain($temp\_cipher$, mask_e)
    load_management("c2p", $temp\_cipher$, relin_keys, evaluator)
    $temp\_cipher \leftarrow$ evaluator.rotate_vector($temp\_cipher$, $-ind$, galois_keys)
    if final_cipher **is nil then**
      final_cipher $\leftarrow temp\_cipher$
    else
      evaluator.add_inplace(final_cipher, $temp\_cipher$)
    end if
    $ind \leftarrow ind + n$
  end for
end for
return final_cipher

Table 9: Matrix transposition performances. Source: Created by the authors.

Matrix transposition	Execution time [s]	Multiplicative depth
Proposed scheme	0.315	1
Library example	0.912	2

Our version is also much faster than the library example when running under the same conditions.

Table 10: Angle Reduction into the Interval. Source: created by the authors.

Algorithm: Angle normalization by wrapping and conditional shifting

Require: Angle in radians `angle`
Ensure: Reduced angle within the interval $\left[-\frac{10\pi}{8}, \frac{10\pi}{8}\right]$ ▷ Step 1: Map the angle to the interval $[0, 2\pi]$
1: $angle \leftarrow angle \mod (2\pi)$
▷ Step 2: Check and adjust to the target interval $\left[-\frac{10\pi}{8}, \frac{10\pi}{8}\right]$
2: **if** $angle > \frac{10\pi}{8}$ **or** $angle < -\frac{10\pi}{8}$ **then**
3: **return** `np.where`$(angle > 0, angle - 2\pi, angle + 2\pi)$
4: **else**
5: **return** $angle$
6: **end if**

5 Evaluation and Results

In this section, we present the results obtained using the complete Extended Kalman Filter (EKF) when operating on encrypted inputs and compare them to those obtained using the unencrypted EKF. The performance evaluation relies on the following metrics:

- The Root Mean Square Error (RMSE) for each component of the estimated state vector,
- The Frobenius norm of the absolute error matrix computed at each time step using the unencrypted EKF estimates as the reference,
- The component-wise relative error of the state vector, also computed at each time step with the unencrypted estimates as reference.

As a reminder, the Frobenius norm of a $m \times n$ matrix $M = A - B$ is defined as follows:

$$\|A - B\|_F = \sqrt{\sum_{1 \le i \le m, 1 \le j \le n} |A_{ij} - B_{ij}|^2} \quad (45)$$

The estimation process was limited to five consecutive time steps, owing to the significant computational overhead introduced by homomorphic encryption. Accordingly, five encrypted state vector estimates were generated and compared to the corresponding unencrypted EKF estimates. On average, the encrypted EKF required at least 16 minutes and 7 seconds to process all five steps. However, running a single time step required a minimum of 3 minutes and 6 seconds.

6 Discussion and Limitations

Implementing homomorphic equivalents of the previously listed critical functions constituted the first step toward obtaining the results. Our objective was to perform these operations using a minimal number of multiplications and within a reasonable computation time to minimize the need for frequent decryptions and re-encryptions and reduce the execution time of the encrypted EKF. This was intended to limit the latency CKKS introduced while preserving its computational security advantage. During this phase, the parameters governing numerical precision in CKKS with SEAL were configured, most notably, inner primes set to 40 bits and outer primes set to 60 bits.

The overall Root Mean Square Error (RMSE) analysis presented in Table 11 indicates a low error level for all state vector components compared to the original filter, whose resilience against eavesdropping attacks we aim to enhance. Notably, the two components directly related to position (x and y) exhibit RMSE values in millimeter order, which reflects a preservation of the original filter's precision. This can also be observed in Figure 4 and Figure 5. Although the relative errors for these two parameters can appear large, as illustrated in Table 12, it should be emphasized that these are geographic coordinates. In contrast, the absolute error between the encrypted and unencrypted filter estimates is independent of the chosen reference frame. However, the analysis of the Frobenius norm evolution over each sampling instant, as presented in Table 13, reveals a progressive increase in the discrepancy between the encrypted and unencrypted filter estimates, reaching a value of 0.59 at the fifth time-step. This trend is also apparent when examining individual component-wise deviations in the state vector estimates at each time step as depicted in Figure 6, Figure 7, Figure 8, and Figure 9, along with the relative error distribution using the unencrypted filter as the reference. In this case, however, a notable nuance appears in the error distribution: the higher deviations observed in the Frobenius norm are mainly driven by the yaw angle. Regarding filter consistency, Figure 10 illustrates a clear trend indicating the preservation of statistical consistency, as reflected by the distribution of NIS values over the first five-time steps, even with the integration of the CKKS homomorphic encryption scheme into the EKF. This result suggests that including encryption does not significantly degrade the filter's performance, which is favorable.

Table 11: Component-wise RMSE. Source: created by the authors.

Parameter	RMSE
x	0.003154
y	0.002116
ψ	0.377384
v	0.043968
a	0.039673
$\dot{\psi}$	0.013821

Only the heading angle stands out with a slightly larger RMSE value.

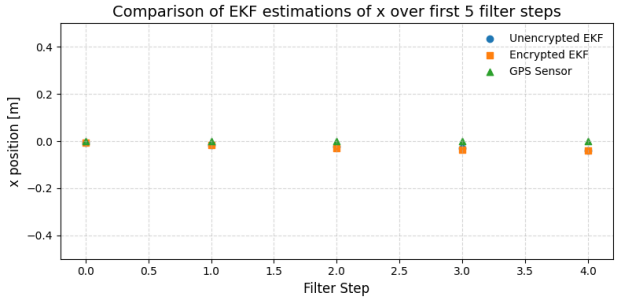

Fig. 4: Position x in encrypted (orange squares) and unencrypted (blue circles) EKF, and the reference from the GPS sensor (green triangles). Source: Created by the authors.

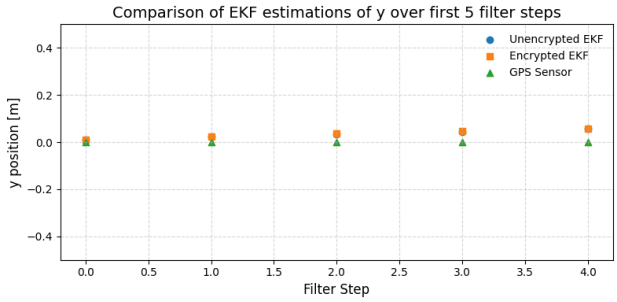

Fig. 5: Position y in encrypted (orange squares) and unencrypted (blue circles) EKF, and the reference from the GPS sensor (green triangles). Source: Created by the authors.

Table 12: Component-wise relative error results. Source: created by the authors.

Time-step	Δx	Δy	$\Delta\psi$	Δv	Δa	$\Delta\dot{\psi}$
1	1.510467	1.267079	4.629315	1.246887	1.233312	1.230573
2	5.089839	1.293983	9.443739	6.876329	24.371044	2.702100
3	23.963513	5.911789	14.837312	8.270551	6.036437	4.254900
4	12.094966	7.920437	21.062731	7.181241	1.199757	5.840101
5	3.993823	4.312367	27.142829	7.048161	3.521703	6.875530

Only the heading angle stands out with an increasingly large relative error value.

Table 13: Frobenius norm of the absolute error on the state vector. Source: created by the authors.

Time-step	$\|\Delta\mathbf{X}\|_F$
1	0.10181
2	0.22220
3	0.33257
4	0.46028
5	0.59029

Computed at each time-step.

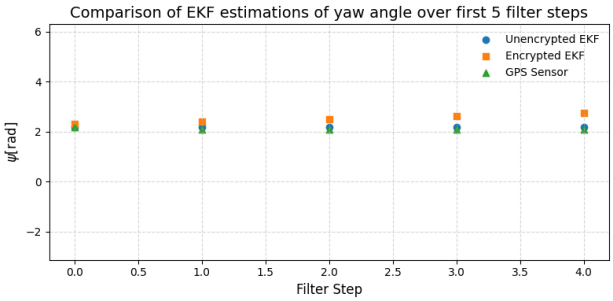

Fig. 6: Yaw angle ψ in encrypted (orange squares) and unencrypted (blue circles) EKF, and the reference from the GPS sensor (green triangles). Source: Created by the authors.

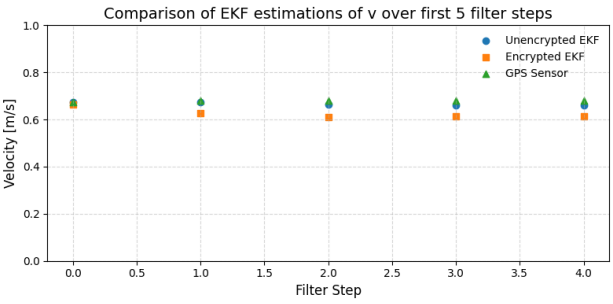

Fig. 7: Velocity v in encrypted (orange squares) and unencrypted (blue circles) EKF, and the reference from the GPS sensor (green triangles). Source: Created by the authors.

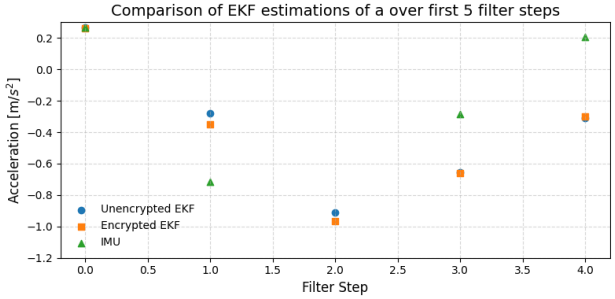

Fig. 8: Linear acceleration a in encrypted (orange squares) and unencrypted (blue circles) EKF, and the reference from the IMU (green triangles). Source: Created by the authors.

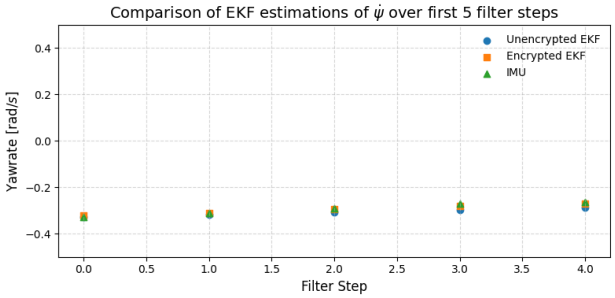

Fig. 9: Yaw rate $\dot{\psi}$ in encrypted (orange squares) and unencrypted (blue circles) EKF, and the reference from the IMU (green triangles). Source: Created by the authors.

The main challenge in our context remains the computational overhead of encrypted operations. Initially, we limited our implementation to a single time step before introducing optimization measures. The most impactful adjustment was the duplication of encryption contexts, which allowed us to avoid wasting computational resources by using lighter contexts for operations that did not require the heavier, more resource-intensive ones, resulting in a significant time gain.
At first, execution times averaged around 15 minutes for a single sampling instant. However, through a series of optimizations described in the filter design section, we reduced the execution time for a single sampling step to as low as 3 minutes and 6 seconds, of which 66.25 seconds were spent on operations in the first context, and 119.94 seconds on operations in the second context. Consequently, computations over five sampling instants were completed in as little as 16 minutes and 7 seconds. However, it is important to note that this remains significantly higher than the unencrypted filter, which performs the same operations under identical conditions in approximately 0.05 seconds.

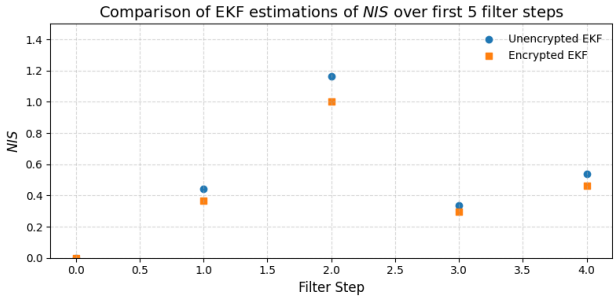

Fig. 10: NIS in encrypted (orange squares) and unencrypted (blue circles) EKF. Source: Created by the authors.

We also observed that the ciphertext replication strategy, while effectively limiting the need for frequent decryptions and re-encryptions caused by ciphertext reuse as function arguments in multiple critical expressions, results in high memory consumption. This could become a limiting factor in long-term deployments.

Finally, with respect to effective security and under the assumption that context-switching operations and the matrix inversion of the innovation covariance are executed within a secure enclave, it is important to note that potential vulnerabilities persist. In particular, if the enclave is compromised, an attacker may still be able to extract the state vector, either during the context-switching phase at the beginning of a time step or during intermediate computations within a given time step.

7 Conclusion and Future Work

This study proposes a significantly more secure Kalman filter-based state estimator. The Extended Kalman Filter (EKF), applied with the Constant Turn Rate and Acceleration (CTRA) motion model in a navigation/tracking context, processes encrypted inputs, performs internal computations, and exchanges information with a secure external entity—without ever accessing the plaintext content of the computations or their results. This enhanced resilience to eavesdropping comes at the cost of increased computational latency and a slight decrease in estimation accuracy. The latter is primarily due to limitations of the employed homomorphic encryption library, which constrained us to use low-degree polynomial approximations for nonlinear functions. The results obtained indicate that, in a context limited to localization—specifically the estimation of x and y coordinates—and where latency constraints can be mitigated, it is feasible to employ a homomorphic encryption scheme such as CKKS to secure a state estimator embedded within a cyber-physical system against eavesdropping attacks.

This approach provides data confidentiality while preserving the estimation accuracy of its unencrypted counterpart. However, extending this approach to broader navigation tasks beyond mere localization would require addressing the slight but accumulating discrepancies observed between the encrypted and unencrypted filters. Over many time-steps, these growing errors may compromise the estimator's long-term reliability.

Considering all these aspects, we believe that future work should focus, if not on improving existing encryption schemes to make them faster, on shifting efforts toward more flexible and higher-performance computing hosts. Two main directions emerge:

- The first involves continuing with our current approach of using a constrained library but deploying it on a host with greater computational power and memory capacity. This, especially if combined with additional algorithmic optimizations, could drastically reduce execution time and help mitigate the impact of high memory usage.
- The second approach involves moving toward hosts that offer greater design flexibility, both in terms of architecture, which could be tailored to the CKKS encryption scheme, and local implementation, which could incorporate additional operations such as bootstrapping. If fast enough, this would optimize our results by reducing memory pressure and eliminating the need for ciphertext replication. Reconfigurable hardware (FPGAs) and application-specific integrated circuits (ASICs) represent reliable alternatives for this hardware-level optimization approach.

References

[1] Rasim Alguliyev, Yadigar Imamverdiyev, and Lyudmila Sukhostat, "Cyber-physical systems and their security issues," *Computers in Industry*, vol. 100, pp. 212–223, 2018. DOI: 10.1016/j.compind.2018.04.017.

[2] Abdul Rahman Sani, Muneeb Ul Hassan, and Jinjun Chen, "Privacy preserving machine learning for electric vehicles: A survey," *arXiv preprint arXiv:2205.08462*, 2022. DOI: 10.48550/arXiv.2205.08462.

[3] Greg Welch and Gary Bishop, *An introduction to the kalman filter*, https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, Access Date: 28/11/2024.

[4] Jean-Paul A Yaacoub, Ola Salman, Hassan N Noura, Nesrine Kaaniche, Ali Chehab, and Mohamad Malli, "Cyber-physical systems security: Limitations, issues and future trends," *Microprocessors and microsystems*, vol. 77, p. 103 201, 2020. DOI: 10.1016/j.micpro.2020.103201.

[5] Mohsen Zamani, Ladan Sadeghikhorrami, Ali Akbar Safavi, and Farhad Farokhi, *Private state estimation for cyber-physical systems using semi-homomorphic encryption*, http://mtns2018.ust.hk/media/files/0078.pdf, Access Date: 28/10/2024.

[6] Junsoo Kim, Chanhwa Lee, Hyungbo Shim, Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016. DOI: 10.1016/j.ifacol.2016.10.392.

[7] Andreea B Alexandru and George J Pappas, "Encrypted lqg using labeled homomorphic encryption," in *Proceedings of the 10th ACM/IEEE international conference on cyber-physical systems*, 2019, pp. 129–140. DOI: 10.1145/3302509.3311049.

[8] Jingyu Wang, Dongyuan Shi, Jinfu Chen, and Chen-Ching Liu, "Privacy-preserving hierarchical state estimation in untrustworthy cloud environments," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1541–1551, 2020. DOI: 10.1109/TSG.2020.3023891.

[9] Ladan Sadeghikhorami and Ali Akbar Safavi, "Secure distributed kalman filter using partially homomorphic encryption," *Journal of the Franklin Institute*, vol. 358, no. 5, pp. 2801–2825, 2021. DOI: 10.1016/j.jfranklin.2020.08.048.

[10] Zhenyong Zhang, Junfeng Wu, David Yau, Peng Cheng, and Jiming Chen, "Secure kalman filter state estimation by partially homomorphic encryption," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2018, pp. 345–346. DOI: 10.1109/ICCPS.2018.00046.

[11] Francisco J Gonzalez-Serrano, Adrian Amor-Martın, and Jorge Casamayon-Anton, "State estimation using an extended kalman filter with privacy-protected observed inputs," in *2014 IEEE international workshop on information forensics and security (WIFS)*, IEEE, 2014,

pp. 54–59. DOI: 10.1109/WIFS.2014.7084303.

[12] William E Curran, Cesar A Rojas, Leonardo Bobadilla, and Dylan A Shell, "Oblivious sensor fusion via secure multi-party combinatorial filter evaluation," in *2021 60th IEEE Conference on Decision and Control (CDC)*, IEEE, 2021, pp. 5620–5627. DOI: 10.1109/CDC45484.2021.9683202.

[13] Huelse, *Seal-python - matrix operations example*, https://github.com/Huelse/SEAL-Python/blob/main/examples/matrix_operations.py, Access Date: 27/12/2024, 2024.

[14] Markus Balzer, *Kalman filter implementation in python*, https://github.com/balzer82/Kalman, Accessed: 28/08/2024, 2023.

[15] Alonzo Kelly, *A 3d state space formulation of a navigation kalman filter for autonomous vehicles*, https://apps.dtic.mil/sti/html/tr/ADA282853/, Accessed: 28/12/2024.

[16] Ian Reid and Hilary Term, *Estimation ii*, https://cs.adelaide.edu.au/users/ianr/Teaching/Estimation/LectureNotes2.pdf, Accessed: 05/12/2024.

[17] Wael Farag, "Self-driving vehicle localization using probabilistic maps and unscented-kalman filters," *International Journal of Intelligent Transportation Systems Research*, vol. 20, no. 3, pp. 623–638, 2022. DOI: 10.1007/s13177-022-00314-4.

[18] Robin Schubert, Christian Adam, Marcus Obst, Norman Mattern, Veit Leonhardt, and Gerd Wanielik, "Empirical evaluation of vehicular models for ego motion estimation," in *2011 IEEE intelligent vehicles symposium (IV)*, IEEE, 2011, pp. 534–539. DOI: 10.1109/IVS.2011.5940526.

[19] Microsoft Research, *Microsoft seal (simple encrypted arithmetic library)*, https://github.com/microsoft/SEAL, Access Date: 27/12/2024, 2013.

[20] Rashmi Agrawal and Ajay Joshi, *On Architecting Fully Homomorphic Encryption-based Computing Systems*. Jan. 2023, ISBN: 978-3-031-31753-8. DOI: 10.1007/978-3-031-31754-5.

[21] Wei Zheng, Ying Wu, Xiaoxue Wu, Chen Feng, Yulei Sui, Xiapu Luo, and Yajin Zhou, "A survey of intel sgx and its applications," *Frontiers of Computer Science*, vol. 15, pp. 1–15, 2021. DOI: 10.1007/s11704-019-9096-y.

[22] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1209–1222. DOI: 10.1145/3243734.3243837.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.