A Trend-following Trading Indicator on Homomorphically Encrypted Data

Haotian Weng[a] and Artem Lenskiy[b]
Research School of Computer Science, The Australian National University, Canberra, Australia

Keywords: Homomorphic Encryption, Quantitative Finance, Algorithmic Trading.

Abstract: Algorithmic trading has dominated the area of quantitative finance for already over a decade. The decisions are made without human intervention using the data provided by brokerage firms and exchanges. An emerging intermediate layer of financial players that are placed in between a broker and algorithmic traders has recently been introduced. The role of this layer is to aggregate market decisions from the algorithmic traders and send a final market order to a broker. In return, the quantitative analysts receive incentives proportional to the correctness of their predictions. In such a setup, the intermediate player — an aggregator — does not provide the market data in plaintext but encrypts it. Encrypting market data prevents quantitative analysts from trading on their own, as well as keeps valuable financial data private. This paper proposes an implementation of a popular trend-following indicator with two different homomorphic encryption libraries — SEAL and HEAAN — and compares it to the trading indicator implemented for plaintext. Then, an attempt to implement a trading strategy is presented and analysed. The trading indicator implemented with SEAL and HEAAN is almost identical to that implemented on the plaintext, with the percentage error of 0.14916% and 0.00020% respectively. Despite many limitations that homomorphic encryption imposes on this algorithm's implementation, quantitative finance has a potential of benefiting from the methods of homomorphic encryption.

1 INTRODUCTION

The most prominent approach that provides the means for data analysis and also keeps the data private is based on homomorphic encryption (HE). The idea of HE that allows computation on encrypted data was firstly proposed in the 1970s. However, no practical implementation existed until Craig Gentry proposed one in his PhD thesis in 2009 (Gentry and Boneh, 2009). Modern cryptosystems are capable of performing arbitrary computation on encrypted data - ciphertexts, facilitating the implementation of various data analysis tools (Aslett et al., 2015). As an active area of research, there is a multitude of HE schemes that have been proposed and implemented as open-source libraries. CKKS (Cheon et al., 2017) and BFV (Brakerski and Vaikuntanathan, 2014) are among the most popular HE schemes. Simple Encrypted Arithmetic Library (SEAL) developed by Microsoft Cryptography Research (SEAL, 2019) implements both BFV and CKKS scheme. Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) is the original name of CKKS scheme developed by the Seoul National University CryptoLab which only supports CKKS scheme. A particular area of HE that has not receive well-deserved attention is the privacy-preserving analysis of time-dependent data. Such topic attracts financial industry's attention where time-series analysis is widely applied.

The financial industry is known to be extremely cautious about privacy aspects of storing, analysing and distributing data. The solutions for secure data storage have been available for decades and are already provided by numerous cloud services. Secure data distribution is nowadays an integral part of the Internet, thanks to Secure Sockets Layer (SSL). However, data analysis that preserves privacy is still in its infancy, even though, as mentioned, it has been actively developed, and adopted by the financial industry.

Numer.ai is a hedge fund powered by thousands of independent quantitative analysts striving to outperform the market (Numer.ai, 2019). The quantitative analysts compete with each other, and those with accurate predictions are rewarded. Numer.ai does not

[a] https://orcid.org/0000-0002-3993-7621
[b] https://orcid.org/0000-0002-4745-6756

Weng, H. and Lenskiy, A.
A Trend-following Trading Indicator on Homomorphically Encrypted Data.
DOI: 10.5220/0009835706020607
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 602-607
ISBN: 978-989-758-446-6

provide the market data in plaintext but transforms it in a form that makes it impossible to know what financial asset a particular time series represents. This, in turn, prevents quantitative analysts from trading on their own and keeps valuable financial data private. To the best of our knowledge, Numer.ai relies on proprietary obfuscation methods.

In our scenario, we distinguish three independent players: a broker, a decision aggregator and algorithmic traders. A broker provides access to the exchanges that is a mediator between sellers and buyers. In particular, it provides such market data as order books, recent trades and price quotes, so the market participants are able to draw a trading decision. The decision aggregator (analogous to Numer.ai) receives market data M in plaintext and returns market orders O. The data received by the aggregator is encrypted as $c_k = \mathcal{E}_1\{m_k\}$ and is sent to an algorithmic trader T_k. Algorithmic traders operate over the encrypted data c_k and the decision o_k^* drawn by the traders are also encrypted and unknown to the traders (fig. 1). These decisions are then decrypted by the aggregator $o_k = \mathcal{D}a_1\{o_k^*\}$ and transmitted to the broker in the form of market orders.

In this paper, we focus on making a very first step in the direction of applying HE in developing an algorithm that operates on encrypted time-dependent data. On the example of a popular trend following trading strategy based on Moving Average Convergence Divergence (MACD) indicator, we demonstrate how an algorithmic trader could employ methods of homomorphic encryption to make trading decisions. [1]

2 BACKGROUND

2.1 Moving Average Convergence Divergence

MACD is a momentum indicator which uses the difference between fast and slow moving averages to indicate market trend (Appel, 1979). MACD was a valuable tool for traders during the 1980s. In this paper, we implement a modified MACD algorithm that operates on encrypted stock price.

2.2 Homomorphic Encryption

In the past decade, several homomorphic encryption schemes have been introduced. One of the most popular schemes is CKKS that implements approximate arithmetic of complex numbers. The scheme supports addition, subtraction and multiplication (Cheon et al., 2017).

Every operation in CKKS, especially multiplication, adds a certain amount of noise, which limits the number of operations allowed before the accumulated noise grows to the point making the final result inaccurate. Levelled schemes limit the maximal quantity of sequential homomorphic multiplications before the noise becomes intolerable (Brakerski and Vaikuntanathan, 2014).

To fight the noise, it introduces rescaling as well as bootstrapping. The rescaling is a scale-invariant technique that scales down the size of ciphertext modulus to reduce the noise and preserve the precision (Cheon et al., 2017). The bootstrapping operation in theory eliminates the noise accumulated throughout homomorphic computations by refreshing the noise in a ciphertext. The significant disadvantages of the bootstrapping method are the dramatic increase in computational time and significant memory consumption (Cheon et al., 2018).

3 RELATED WORK

Previous studies have shown the practicality of privacy-preserving analysis of time-series data. An additive homomorphic encryption scheme was proposed to aggregate time-series data without sacrificing privacy (Shi et al., 2011). The ciphertexts are encrypted under different users' secret keys respectively to achieve secure multi-party computation. Additionally, Paillier encryption scheme as a partial homomorphic encryption was applied to privacy-preserving similarity evaluation of time-series data (Zhu et al., 2014). Pallier scheme is adequate for computing the square of Euclidean distance as it supports homomorphic addition of ciphertexts and homomorphic multiplication to plaintexts. Another partially homomorphic-encryption-based access control construction (HEAC) was introduced to support both access control and aggregation-based computations on encrypted data (Burkhalter et al., 2020). However, these studies rely on additive homomorphic encryption schemes and thus the types of computations allowed are limited. Our approach focuses on financial time-series data and applies both homomorphic addition and homomorphic multiplication of ciphertexts to generate the trading decisions.

[1] https://github.com/woonhulktin/HETSA

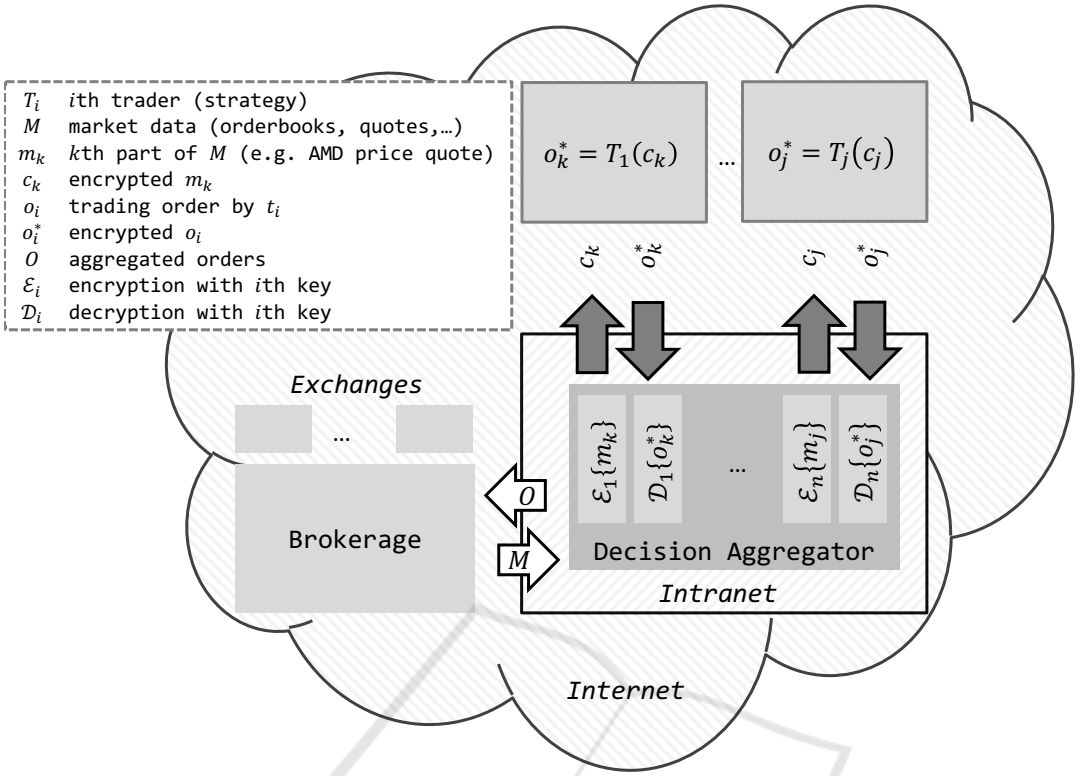

Figure 1: An intermediate layer collects encrypted decision, decrypts them and makes a final decision on whether to buy or to sell.

4 METHODS

4.1 Data Encoding

Both SEAL and HEAAN libraries process data in batches. This, in turn, speeds up the processing since all components of the vector are processed simultaneously (Chen et al., 2017). However, in our scenario, the server sends encrypted price one at a time, and the remaining components are padded with zeros. This process repeats every time a new price quote is available. Therefore, only the first slot of the plaintext vector is used to store the asset price. This results in higher memory requirement and slows computation.

Once receiving the latest encrypted price quote, the algorithm appends it to the vector of previously received ciphertexts, and a MACD signal value is produced, that is either used to generate a trading decision or returned on its own.

As the share prices are rational numbers and BFV is slower than CKKS when performing multiplication followed by rescaling (Chen et al., 2019), the fractional encoder is chosen over the integer one.

To test our algorithm on encrypted data, we selected Apple's daily stock price (NASDAQ: AAPL) on the interval from 06/01/2015 to 21/10/2015.

4.2 Weighted Moving Average

A moving average filter is a low pass filter with a linear phase shift. In the context of financial time-series, it is used to determine a trend of an asset price and is the foundation of the MACD indicator and corresponding trading strategy. The emphasis is put on recent price quotes by assigning different weighting factors to the asset prices. The original MACD indicator employs the exponential moving average (EMA), and as a first-order autoregressive filter, it requires recursion. Both SEAL and HEAAN libraries are limited by the noise that accumulates with every arithmetic operation. Without an efficient bootstrapping method, EMA is infeasible as it requires a significant number of multiplicative depths (in theory infinitely many). One solution is to replace the EMA by a non-recursive filter. One such filter is the weighted moving average (WMA). The WMA has a finite impulse response and does not require infinite multiplications. Instead, it limits the multiplicative depth by the order n that defines the number of multiplications and is

equal to the window size. The weighting coefficients of the WMA are chosen as follows:

$$\boldsymbol{w}[i] = \frac{2(i+1)}{n(n+1)} \quad (1)$$

for $i \in [0,n)$, where $\boldsymbol{w}[i]$ is the weight and n is the window size.

Algorithm 1: Weighted Moving Average (WMA).

Input: a vector of ciphertexts $\boldsymbol{c}$, window size n
Output: a vector of encrypted weighted moving averages $\boldsymbol{a}$

```
w = zeros(n)
for i in range [0, n) do
    w[i] = FHE.encode(2(i+1)/(n(n+1)))
end for
a = zeros(c.size − n)
for i in range [0, c.size − n) do
    a[i] = FHE.encrypt(0)
    for j in range [0, n) do
        r = FHE.multiplyConstant(c_{i:i+n}[j], w[j])
        a[i] = FHE.add(a[i], r)
    end for
end for
return a
```

4.3 Moving Average Convergence Divergence

One of the popular indicators for detecting a market turning point is the MACD indicator (Appel, 2003). The original MACD indicator computes two moving averages: the 12-period EMA and the 26-period EMA. Both EMAs are replaced by the WMAs for the reasons explained above.

The trading signals are triggered by the MACD signal line crossing the x-axis. First, a 12-period WMA $\boldsymbol{\alpha}$ and a 26-period WMA $\boldsymbol{\beta}$ are computed. Then, the differences between $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are calculated. Finally, a 9-period WMA $\boldsymbol{\gamma}$ is applied to the differences to produce the MACD signal line $\boldsymbol{m}$. Algorithm 2 illustrates a library-independent MACD implementation using homomorphic encryption method.

When the MACD signal line crosses the x-axis from below, it indicates a buy signal and when the signal line crosses the x-axis from above, it triggers a sell signal.

4.4 Trading Decision

Unfortunately due to the theoretical limitations of CKKS as well as other popular HE schemes, only basic arithmetic operations are provided. None of

Algorithm 2 : Moving Average Convergence Divergence Function (MACD).

Input: a vector of ciphertexts of asset prices $\boldsymbol{d}$
Output: a vector of ciphertexts of MACD signals $\boldsymbol{m}$

```
α = wma(d, 12)
β = wma(d, 26)
θ = zeros(β.size)
for i in range[0, β.size) do
    θ[i] = FHE.sub(α_{14:α.size}[i], β[i])
end for
γ = wma(θ, 9)
m = zeros(γ.size)
for i in range[0, γ.size) do
    m[i] = FHE.sub(θ_{9:θ.size}[i], γ[i])
end for
return m
```

widespread used HE libraries is equipped with logic and relational operators on numbers (Acar et al., 2018). Hence, there is no direct method to compare two values and deduce where the MACD signal line is greater than, less than, or equal to zero which in turn determines the time of buying, selling or doing nothing. Nevertheless, the decision of sell, hold, or buy could be associated with a decision function o that produces either -1, 0 or 1 that correspond to a sell, hold or buy order. To define a decision function o, we first define a *sign* function that return the sign of a number. Then we define a vector of differences of adjacent MACD values $\boldsymbol{\delta}$ as $\boldsymbol{\delta}[i] = \boldsymbol{m}[i-1] - \boldsymbol{m}[i]$ with $\boldsymbol{\delta}[0] = 0$, and the product of consecutive MACD values $\boldsymbol{\pi}$ as $\boldsymbol{\pi}[i] = \boldsymbol{m}[i-1]\boldsymbol{m}[i]$ for $i \in [1, \boldsymbol{m}.size)$ with $\boldsymbol{\pi}[0] = 0$. Then the trading decision is defined as:

$$o_1(\boldsymbol{m}, i) = \frac{1}{2} sign(\boldsymbol{\delta}[i]) \cdot (sign(\boldsymbol{\pi}[i]) - 1) \quad (2)$$

where $i \in [0, \boldsymbol{m}.size)$.

The range of the function above is $\{-1, 0, 1\}$. The downside of the function is its dependence on the *sign* function that is not implementable using available HE operations. The first *sign* function determines the trend change and the second is the moment of crossing x-axis.

4.5 Polynomial Approximation of the ReLU Function

Given that HE schemes are limited to arithmetic operations, only polynomial functions can be implemented homomorphically. Additionally, due to the noise accumulation discussed earlier, there is a limitation on the order of a polynomial function. From

this perspective, due to discontinuity of the *sign* function, polynomials of a lower order do not approximate it well. A better function in terms of polynomial approximation is the ReLU function that is defined as follows:

$$r(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \quad (3)$$

Then an equivalent to o_1 decision function could be implemented using ReLU as follows:

$$o_2(\boldsymbol{m}, i) = -sign(\boldsymbol{\delta}[i] \cdot r(-\boldsymbol{\pi}[i])) \quad (4)$$

where $i \in [0, \boldsymbol{m}.size)$.

We employed polynomial regression to approximate the ReLU function to estimate the polynomial coefficients:

$$\begin{aligned} \hat{r}(x) = &-0.0001x^9 - 0.0003x^8 + 0.0025x^7 \\ &+ 0.009x^6 - 0.0253x^5 - 0.0984x^4 + 0.0882x^3 \\ &+ 0.5173x^2 + 0.4475x + 0.0753 \end{aligned} \quad (5)$$

Then using the $\hat{r}$ we define an approximation of the decision function as follows:

$$\hat{o}_2(\boldsymbol{m}, i) = -\boldsymbol{\delta}[i] \cdot \hat{r}(-\boldsymbol{\pi}[i]) \quad (6)$$

where $i \in [0, \boldsymbol{m}.size)$.

We apply both o_2 and $\hat{o}_2$ on plaintext MACD signals and employ only $\hat{o}_2$ on encrypted MACD signals. The percentage errors between plaintext and ciphertext implementations of $\hat{o}_2$ are then compared and presented in Section 4.

4.6 Multiplicative Depth

Multiplicative depth is the maximal number of sequential homomorphic multiplications allowed, while the multiplication level represents the number of sequential multiplications performed on the ciphertext (Brakerski and Vaikuntanathan, 2014). Namely, a polynomial's multiplicative depth depends on its degree. Multiplicative depth is defined by the parameters of the HE library. Every time a multiplication operation is performed on a ciphertext, its multiplication level goes one level deeper. The number of sequential multiplications on the ciphertext are limited by the multiplicative depth. The most substantial part of the trading strategy in terms of multiplicative depth is the implementation of $\hat{r}$ function. A multiplicative depth of 3 is required to generate $\boldsymbol{m}$ but 7 more levels are needed to produce $\boldsymbol{\delta}$. Currently, $\hat{r}$ is of degree 8 and consumes 4 multiplication levels. The trading decision $\hat{o}_2$ is of degree 9. Therefore the maximum multiplication level of our approach is 9.

4.7 Confidentiality

CKKS scheme is an asymmetric cryptosystem with a public and a private keys. The public key is shared with both the decision aggregator and algorithmic traders while the private key is shared with the decision aggregator. Without knowing the private key, algorithmic traders as well as the attackers who hijack the communication are not able to decrypt the ciphertexts. Therefore the confidentiality of our method is well maintained by CKKS scheme. Moreover, every trader is equipped with a unique public key, that serves as a digital sign and prevents traders from impersonating each other.

5 RESULTS

To compare the errors between the WMA-based ciphertext and plaintext implementations, we employ the mean absolute percentage error defined as:

$$e(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} \frac{|\boldsymbol{x}[i] - \boldsymbol{y}[i]|}{\boldsymbol{y}[i]} \times 100\% \quad (7)$$

where $i \in [0, \boldsymbol{x}.size)$ and $N = \boldsymbol{x}.size = \boldsymbol{y}.size$. $\boldsymbol{x}$ is either decrypted WMA, MACD signals or the trading decisions and $\boldsymbol{y}$ is the vector of corresponding plaintext WMA, MACD signals or trading decisions.

Table 2 presents the comparison results. As the errors of WMA and MACD signals between encrypted and plaintext analysis are insignificant, the WMA and MACD signals generated with SEAL and HEAAN are almost identical to those in plaintext. In terms of trading decisions, we compare trading decision functions over encrypted data with SEAL and plaintext data. The peaks of the approximated trading decisions generally correspond to the exact trading decisions, but there are errors around the peaks. Additionally, the error increases as the multiplication level gets deeper, and hence the percentage error of a trading decision function is larger than the that of MACD and much larger than that of WMA. The reason for the increasing error is the noise added by every arithmetic operation.

Table 1: Errors between ciphertext and plaintext analysis.

Result	Percentage Error
WMA-SEAL	0.00918%
WMA-HEAAN	0.00019%
MACD-SEAL	0.14916%
MACD-HEAAN	0.00020%
Decision-SEAL	3.19030%
Decision-HEAAN	0.03794%

The computation was conducted on the Intel(R) Core(R) CPU i7-7820HQ @ 2.9GHz with 16GB RAM. The task consisted of the following steps: (1) 200 AAPL share prices were encrypted, (2) MACD analysis was performed, (3) encrypted trading decisions were generated and (4) the decisions were decrypted. The time required to produce the MACD signal as well as the trading decision for a single day were measured and summarised in table 3. The trading indicator implemented with SEAL can also run on 1-second candles and HEAAN implementation can be applied to 15-seconds candles. The total time consumed per data unit is 0.99 and 12.34 seconds for SEAL and HEAAN implementations respectively.

Table 2: Performance of MACD and decision analysis.

Method	Computation Time
MACD-SEAL	0.73 sec
MACD-HEAAN	7.085 sec
Decision-SEAL	0.26 sec
Decision-HEAAN	5.255 sec
Total-SEAL	0.99 sec
Total-HEAAN	12.34 sec

At this point, there are two significant limitations in our implementation. Firstly, the lack of recursion reduces the decision accuracy as the multiplicative depth is constrained by the HE parameters. Although, HEAAN supports bootstrapping, it introduces substantial noise and is also computationally intensive (Acar et al., 2018). Therefore, we did not implement bootstrapping in our approach. Secondly, operation of bootstrapping is slow, due to the absence of logical and relational operators in the state-of-the-art HE libraries. Only approximate decisions can be implemented. However, the presented algorithm accurately implements the MACD indicator and could be applied in the real world applications.

6 CONCLUSIONS

We have implemented the MACD indicator on a stock price time-series. To the best of our knowledge, this is the first time HE methods are applied to financial time-series analysis. The algorithm implemented with SEAL is able to produce the trading indicator in less than a second and could be applied to 1-second candle data as well as to lower resolution data. For the future work we plan to implement linear systems and corresponding recessive AR and MA based filters, including exponential moving average.

REFERENCES

Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35.

Appel, G. (1979). The moving average convergence-divergence method. *Great Neck, NY: Signalert*, pages 1647–1691.

Appel, G. (2003). Become your own technical analyst: How to identify significant market turning points using the moving average convergence-divergence indicator or macd. *The Journal of Wealth Management*, 6(1):27–36.

Aslett, L. J., Esperança, P. M., and Holmes, C. C. (2015). A review of homomorphic encryption and software tools for encrypted statistical machine learning. *arXiv preprint arXiv:1508.06574*.

Brakerski, Z. and Vaikuntanathan, V. (2014). Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871.

Burkhalter, L., Hithnawi, A., Viand, A., Shafagh, H., and Ratnasamy, S. (2020). Timecrypt: Encrypted data stream processing at scale with cryptographic access control. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 835–850.

Chen, H., Dai, W., Kim, M., and Song, Y. (2019). Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 395–412.

Chen, H., Laine, K., and Player, R. (2017). Simple encrypted arithmetic library-seal v2. 1. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer.

Cheon, J. H., Han, K., Kim, A., Kim, M., and Song, Y. (2018). Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 360–384. Springer.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer.

Gentry, C. and Boneh, D. (2009). *A fully homomorphic encryption scheme*, volume 20. Stanford: Stanford university.

Numer.ai (2019). The hardest data science tournament on the planet. $1000000 paid out. https://numer.ai/. Accessed on: 2020-02-14.

SEAL (2019). Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA.

Shi, E., Chan, T. H., Rieffel, E., Chow, R., and Song, D. (2011). Privacy-preserving aggregation of time-series data. In *Proc. NDSS*, volume 2, pages 1–17. Citeseer.

Zhu, H., Meng, X., and Kollios, G. (2014). Privacy preserving similarity evaluation of time series data. In *EDBT*, pages 499–510.