ITUPublications

International Telecommunication Union
Standardization Sector

ITU-T Technical Report

(09/2023)

TR.sgfdm

FHE-based data collaboration in machine learning

Technical Report ITU-T TR.sgfdm

FHE-based data collaboration in machine learning

Summary

Technical Report ITU-T TR.sgfdm provides a guideline for secure data aggregation in machine learning (ML) while protecting input data. It focuses on how fully homomorphic encryption (FHE) works on data aggregations in machine learning. It first describes a general workflow on secure aggregation in ML and explains how FHE-based data aggregation in ML could satisfy a certain requirement. A general workflow is then given on FHE-based ML supporting data aggregation between more than two parties.

Keywords

Data aggregation, data protection, fully homomorphic encryption, machine learning.

Table of Contents

Technical Report ITU-T TR.sgfdm

FHE-based data collaboration in machine learning

1 Scope

This Technical Report provides guidelines for secure data aggregation in machine learning (ML) inference and training using fully homomorphic encryption (FHE) as a building block. It describes a general workflow on data aggregation in ML with a set of security requirements and explains how FHE-based data aggregation in ML could satisfy a certain requirement. FHE primitive itself is out of scope for this Technical Report, but its brief description is given. Finally, a workflow is described on FHE-based ML supporting data aggregation between more than two parties.

2 References

None.

3 Definitions

3.1 Terms defined elsewhere

This Technical Report uses the following terms defined elsewhere:

3.1.1 aggregated data [b-ISO/IEC 20889]: Data representing a group of data principals (see below clause 3.1.3), such as a collection of statistical properties of that group.

3.1.2 attribute [b-ISO/IEC 20889]: Inherent characteristic.

3.1.3 data principal [b-ISO/IEC 20889]: Entity to which data relates.

NOTE – The term "data principal" is broader than "PII principal" (or "data subject" as used elsewhere) and is able to denote any entity such as a person, an organization, a device, or a software application.

3.1.4 dataset [b-ISO/IEC 20889]: Collection of data.

3.1.5 inference [b-ISO/IEC 20889]: Act of deducing otherwise unknown information with non-negligible probability, using the values of one or more attributes (3.1.2) or by correlating external data sources.

3.1.6 microdata [b-ISO/IEC 20889]: Dataset (3.1.4) comprised of records (3.1.7) related to individual data principals (3.1.3).

3.1.7 record [b-ISO/IEC 20889]: Set of attributes (3.1.2) concerning a single data principal (3.1.3).

3.2 Terms defined in this Technical Report

This Technical Report defines the following terms:

3.2.1 computing party: Entity that performs data aggregation (see clause 3.2.2) from multiple input parties (3.2.4), and then statistical analysis and/or machine learning requested by an output party (3.2.6).

3.2.2 data aggregation: Act whereby dataset (3.1.4) is gathered from multiple input parties (3.2.4) to obtain statistical properties or perform machine learning on collected datasets.

3.2.3 input data: Microdata provided by input party (3.2.4).

3.2.4 input party: Entity that provides microdata (3.1.6) for data aggregation (3.2.2).

3.2.5 input data protection: Property that makes sure that the input data (3.2.3) as well as its intermediate results are not disclosed to any other party than the party who provides the input data during data aggregation (3.2.2).

3.2.6 output party: Entity that obtains the computing result on input data (3.2.3) from multiple input parties (3.2.4).

3.2.7 output data protection: Property that prevents data leakage from input data (3.2.3) when the computing result on input data is published to the output party (3.2.6).

4 Abbreviations and acronyms

This Technical Report uses the following abbreviations and acronyms.

AES	Advanced Encryption Standard
AI	Artificial Intelligence
FHE	Fully Homomorphic Encryption
HE	Homomorphic Encryption
ML	Machine Learning
MPC	Multi Party Computation
PET	Privacy Enhancing Technologies

5 Overview

Data aggregation has been arranged either directly between concerned organizations or through trusted third parties with a commitment to certain terms of use. Such conventional methodologies involve manual processes and thus takes considerable resources and time. As data sharing becomes more widespread, data aggregation services are likely to be provided by shared infrastructures with greater effectiveness and efficiency, but such services can potentially be untrusted. For example, during data aggregations, a third party that collects the dataset may intentionally or unintentionally leak the data, and, when anonymous datasets are sequentially combined, data principles can be re-identified [b-Narayanan]. Furthermore, such a trusted party may not be available when data aggregation takes place across borders. It is thus essential to explore ways to achieve secure aggregation under "untrusted" assumptions.

We now provide a general workflow and security requirements considering data aggregation services is provided by shared and untrusted computing infrastructure. There are more than two input parties which provide input data for ML inference or training, and the output party receives the computing result and could be one of the input parties. The computing party then provides the necessary measures for input data protection and output data protection. Input data protection ensures that the input data as well as its intermediate results are not disclosed to any other party than the party that provides the input data. For example, the input party applies the necessary security techniques such as a fully homomorphic encryption (FHE) prior to sending its data to the computing party, and the computing party supports services including FHE-based ML computations while providing the input data protection. There exist cases of using FHE-based data aggregation in ML, and such case studies in official statistics, healthcare, and finance are described in Appendix III. With necessary security controls such as differential privacy in place, output data protection also prevents data leakage from input data when the computing result is published to the output party.

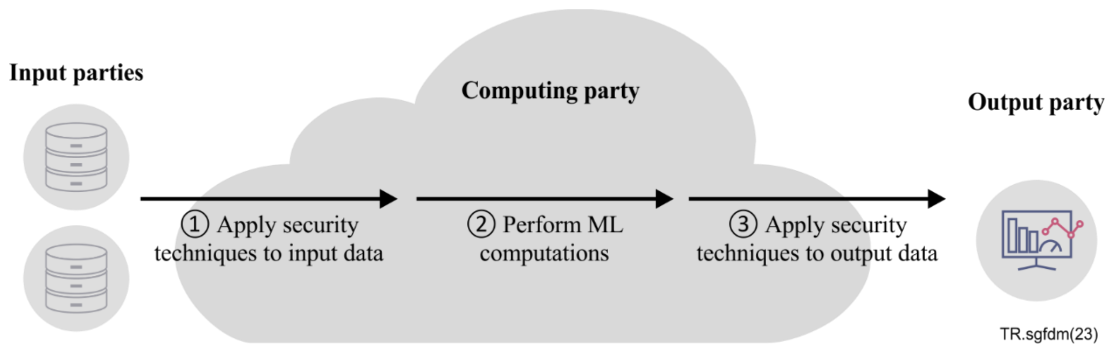

Input parties
Computing party
Output party
① Apply security techniques to input data
② Perform ML computations
③ Apply security techniques to output data
TR.sgfdm(23)

Figure 1 – Overview on data-protecting ML workflow

This guideline focuses on the use of FHE to support input data protection when more than two parties collaborate with their data as input to perform ML computations using 3rd party shared infrastructure services. As mentioned earlier, FHE primitives or schemes are not in the scope of this Technical Report. This guideline however, aims to provide the security guidelines for FHE-based ML supporting data aggregation.

6 FHE as a building block

Cryptography has been extensively studied to secure data-in-transit and data-at-rest with standardized primitives such as RSA [b-PKCS], ECDSA [b-FIPS 186-5], the advanced encryption standard (AES) [b-FIPS 197], etc. Protection for data-in-use, however, becomes increasingly important with more requirements of data sharing and adopting 3rd party computing resources such as cloud computing services. As cryptographic primitives secure data-in-use, homomorphic encryption allows one to perform computations on an encrypted form of data, i.e., ciphertext, without having to first decrypt it for further operations. As shown in Figure 2, when the computation result on the ciphertext is decrypted, it is identical to the output of the same computations on the original data, i.e., plaintext.

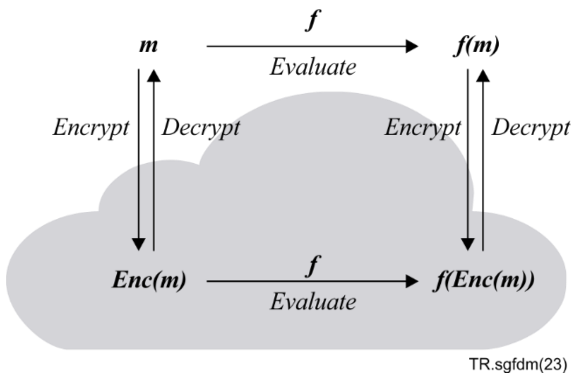

$$\begin{tikzcd} m \arrow[r, "f", "Evaluate"'] \arrow[d, shift right, "Encrypt"'] & f(m) \arrow[d, shift right, "Encrypt"'] \\ Enc(m) \arrow[r, "f", "Evaluate"'] \arrow[u, shift right, "Decrypt"'] & f(Enc(m)) \arrow[u, shift right, "Decrypt"'] \end{tikzcd}$$

Figure 2 – Homomorphic encryption explained

In fact, conventional standardized cryptographic primitives, such as RSA, ElGamal [b-ElGamal], and Paillier [b-Paillier], allows to perform either addition or multiplication in ciphertext; these types of cryptographic primitives are called partial homomorphic encryption. For example, RSA and ElGamal cryptosystems support an unbounded number of modular multiplications on ciphertext, and the Paillier cryptosystem supports an unbounded number of modular additions on ciphertext. The list of related standards is in Appendix I.

There exist yet other types of homomorphic encryption (HE) primitives, called somewhat HE or levelled HE, which support both addition and multiplication on ciphertext, but the numbers of the operations are limited. Gentry [b-Gentry] proposed the notion of FHE by adding to somewhat HE a process called bootstrapping, which renews the remaining number of multiplications. Since Gentry suggested FHE, several FHE primitives have been proposed such as BGV, BFV, TFHE and CKKS. Each FHE primitives are not described in this Technical Report, but the list of primitives and libraries is found in Appendix II.

Since FHE allows arbitrary operations on ciphertext, it thus becomes possible to evaluate functions in machine learning on encrypted data without decrypting them. This could be one of the methods to support input data protection without having to rely on the due diligence of outsourced computing parties. For a practical deployment of FHE as building blocks, however, other building blocks need to be considered such as compliers and secure application framework. First, a lack of knowledge on FHE prohibits engineers to adopt in the field, because they have to consider cryptographic details such as circuit depth analysis, noise tracking, bootstrapping, and cryptographic parameter selection. FHE compliers or also called transpilers, enable to convert existing code that works on plaintext to work on ciphertext generated by the FHE. Second, more importantly, it is not straightforward to use FHE in applications such as data aggregation in ML. It is thus important to construct a secure application framework such as security guidelines for FHE-based data aggregation in ML.

7 Workflows for FHE-based input data protection in ML

FHE could provide input data protection when two input parties provide microdata to the computing party for ML inference. Assume that a data owner wants to utilize a well-trained machine learning model provided by an external organization, such as cancer prediction models with a genome data as input. The genome data is of course sensitive and private information, but the model parameters are also valuable assets to share outside the organization. In this case, the two input parties encrypt their input, i.e., genome data and model parameters, using FHE and then sends them to the computing party. The owner of the genome data, as an output party, could receive the result after the computing party performs inference on those ciphertexts and each party collaborates for decryption. Throughout the whole process, the computing party cannot access both the input and output of the prediction services.

FHE can also be applied to the case where two organizations need to aggregate their data for ML training. Each data set from the input parties could be encrypted using FHE and then computed by the computing party, and the encrypted result can be securely decrypted by a cooperation between the input parties.

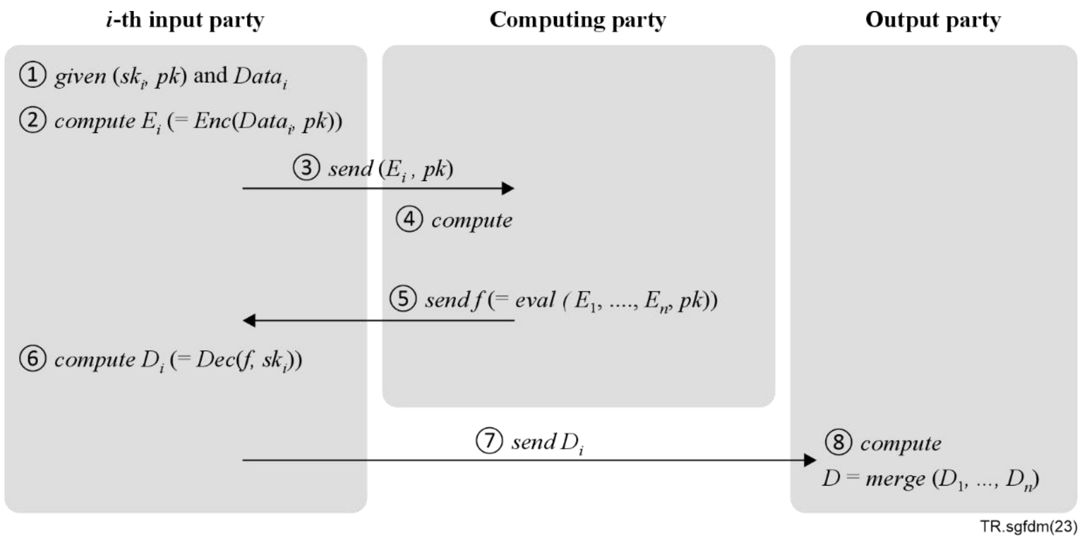
Although FHE allows arbitrary computations such as machine learning (ML) in encrypted data, its use for data aggregation is not straightforward. This clause provides a secure workflow, where more than two parties aggregate their data using FHE for ML inference and training while providing input data protection. There may be more than two input parties, and the output party could be one of the input parties. Input parties aim to aggregate their data either to the ML inference or ML training. The computing party provides FHE-based ML operations, and the output party receives the result, i.e., an inference result or the model parameters of a trained ML model.

We now provide secure workflows in data aggregation for FHE-based ML, as shown in Figure 3.

1) Each input party generates a key pair (sk_i, pk) using distributed secret sharing protocols.
2) Each input party computes the ciphertext E_i $(= Enc(Data_i, pk))$ with its $Data_i$ and public key pk as input.
3) Each input party sends the ciphertext and public key $(E_i,\ pk)$ to the computing party.
4) On receiving the ciphertext and public key from each input party, the computing party performs the operation $eval$, e.g., ML inference or ML training, with the ciphertexts and public key as input.
5) The computing party sends back the result $f(= eval(E_1, \ldots, E_n,\ pk))$ to each input party.
6) Each input party decrypts and generates a decryption share $D_i(= Dec(f,\ sk_i))$.
7) Each input party sends the decryption share to the output party.

8) The output party merge and recover the result, i.e., the inference result of ML or the parameters of the trained ML model.

It is also possible to use multi-key FHE for the case above, where each input parties independently generate and use their key pairs (sk_i, pk_i). The workflow of such a case is the same as Figure 3 except that output parties may interact with input parties to recover the result depending on the underlying FHE primitives. Input parties may also interact with the computing party using cryptographic protocols to accelerate $eval$ depending on the details of ML and the underlying FHE primitives.



i-th input party
Computing party
Output party
① given (sk_i, pk) and $Data_i$
② compute E_i (= $Enc(Data_i, pk)$)
③ send (E_i, pk)
④ compute
⑤ send f (= $eval(E_1, ..., E_n, pk)$)
⑥ compute D_i (= $Dec(f, sk_i)$)
⑦ send D_i
⑧ compute $D = merge(D_1, ..., D_n)$
TR.sgfdm(23)

Figure 3 – A workflow in FHE-based ML

Appendix I

Related standardizations on machine learnings and homomorphic encryptions

The following are important references considered at the beginning of this Technical Report. [b-IEEE 3652.1] is a guide to using machine learning from various input sources, which is the case of ours without the FHE method. ISO/IEC 18033 series are standards for encryption methods including FHE. [b-ISO/IEC 18033-6] defines partial homomorphic encryption (HE) tools supporting single operations in encrypted form, and they can replace FHE in this guideline if an application needs only one type of encrypted operations.

I.1) [b-IEEE 3652.1] *IEEE guide for architecture framework and application of federated machine learning*

- Data aggregation in machine learning is a special case of federated machine learning. Before applying the FHE technique, the base machine learning is assumed to follow the standard.
- Federated machine learning is designed for special machine learning purposes. But FHE can be used not only in machine learning areas but also in any other non-machine learning areas.
- In federated machine learning, not only the final result but also the intermediate parameters of each iteration are revealed; many federated machine learning methods require a trusted third party as client parameters are leaked to the third party. In FHE, there are no intermediate data revealed except the final computation result, and a trusted third party is not needed.

I.2) [b-ISO/IEC 18033-6]:2019 *IT Security techniques – Encryption algorithms – Part 6: Homomorphic encryption*

- In this standard, there are two mechanisms for partially homomorphic encryption: exponential ElGamal encryption and Paillier encryption. The partially homomorphic encryption encompasses schemes that support the evaluation of only one type of operation, e.g., addition (in Paillier encryption) or multiplication (in exponential ElGamal encryption). But FHE can support arbitrary operations and allow arbitrary computation on encryption data.
- If a data aggregation needs only one type of operation, then the encryption mechanism can be applied by partially homomorphic encryption following the standard, rather than the FHE.

Appendix II

FHE libraries and supporting schemes

An FHE library refers to a software package or framework that provides support for a fully homomorphic encryption operation. An FHE scheme is a cryptographic system that can perform several classes of computations on encrypted data, including addition and multiplication. Table II.1 is a collection of popular FHE libraries. The libraries support different FHE schemes. They aim to simplify the development and deployment of FHE applications by providing efficient and secure implementations of FHE schemes and algorithms.

Table II.1 – FHE libraries

Libraries	Supporting FHE schemes
Concrete[b-Concrete]	TFHE [b-Chillotti]
FV-NFLlib [b-FV-NFL]	BFV [b-Brakerski]
FHEW [b-FHEW]	FHEW [b-Ducas]
HEAAN [b-HEAAN]	CKKS [b-Cheon]
HElib [b-HElib]	BGV [b-Brakerski, Z], CKKS [b-Cheon]
Lattigo [b-Lattigo]	BFV [b-Brakerski], CKKS [b-Cheon]
OpenFHE [b-PALISADE]	BFV [b-Brakerski], BGV [b-Brakerski, Z], CKKS [b-Cheon], etc.
SEAL [b-SEAL]	BFV [b-Brakerski], CKKS [b-Cheon]

- The libraries are just examples: each library has no significant difference yet on the performance.
- Language in which each library is built: Concrete: Rust FV-NFLlib, FHEW, HEAAN, HElib, OpenFHE, SEAL: C++ Lattigo: Go
- Characteristics of each FHE scheme: BGV, BFV: advantage in single operation with encrypted integral data

  CKKS: supporting more kinds of operations with encrypted real valued data

  TFHE: less bootstrapping with encrypted data in bit

  FHEW: faster bootstrapping than BGV, BFV with encrypted integral data

Appendix III

Case studies for FHE-based data aggregation

III.1 Official statistics

Nation policy decisions could benefit from data sharing with other countries, but cross-border data sharing is often infeasible because of privacy concerns. The privacy preserving techniques task team (PPTTT) has been active under the statistics division of the United Nations (UN) since April 2018 to advise the UN Committee of experts on big data and data science for official statistics (UN-CEBD), developing the guidelines such as the UN privacy preserving techniques handbook [b-UNH]. The guideline motivates to use privacy-preserving techniques for statistical analysis of sensitive data and present use cases. Those techniques include FHE as well as secure multi-party computation (SMPC), zero-knowledge proof (ZKP), trusted execution environment (TEE), and differential privacy. The task team has also launched the UN PET Lab project in 2020, where multiple national statistics offices (NSOs), such as the United States of America (U.S.) Census Bureau, Statistics Netherlands, the Italian National Institute of Statistics (ISTAT), and the United Kingdom's (U.K.) Office for National Statistics, demonstrate the sharing of sensitive data using the privacy enhancing technologies (PETs).

III.2 Healthcare and finance

One of the interesting efforts to enhance privacy in healthcare is the iDASH privacy & security workshop [b-iDash]. The computation requirements from the growth of genome data enforce to consider more cost-effective cloud computing services, but security and privacy remains a major concern. This community efforts evaluates the performance of various state-of-the-art privacy enhancing technologies. The competition has particularly focused on FHE-related tasks every year since 2015, and challenges regarding FHE-based ML is as follows.

- Homomorphic encryption based logistic regression model learning; to build a machine learning model over data encrypted using FHE.
- Secure multi-label tumour classification using homomorphic encryption; to develop an FHE-based multi-label classification method to predict the classes of tumour samples on genomic information.
- Homomorphic encryption-based secure viral strain classification; to develop FHE-based methods to classify a given viral genome, e.g., COVID-19 genome, into one of the four different strains.
- Secure model evaluation on homomorphically encrypted genotype data; to develop an FHE-based method to securely predict phenotype while protecting both model parameters and genotype.

The World Economic Forum and Deloitte released a white paper [b-WEF] highlighting FHE as one of the five key privacy enhancing technologies (PETs), enabling financial institutions to improve collaboration in 2019. In the same year, there was the global AML and financial crime techSprint [b-FCA], which focused on how encryption privacy enhancing technologies (PETs) including FHE can facilitate the sharing of data in order to tackle money laundering and financial crime concerns, which are related to 800 000 people trafficking and 40 million people under a form of modern slavery every year.

In July 2022, the U.S. and the U.K. governments launched a set of challenges in privacy-enhancing technologies to tackle financial crime and public healthcare issues. The competition proceeds with two separate tracks, i.e., improving the detection of financial crime and bolstering pandemic response capabilities, motivating to develop privacy-preserving federated learning solutions, where artificial intelligence (AI) or ML models are trained on sensitive data without organizations having to share their raw data. The first track aiming for financial crime prevention aims to use PETs for privacy-preserving sharing of financial information, e.g., SWIFT, in order to identify anomalous payments without compromising individual privacy. The second track aims to forecast an individual risk of infection during a pandemic using the dataset representing a digital twin of a regional population.

Bibliography

[b-IEEE 3652.1]	IEEE 3652.1-2020, *IEEE Guide for Architectural Framework and Application of Federated Machine Learning*. <https://standards.ieee.org/ieee/3652.1/7453/>
[b-ISO/IEC 18033-6]	ISO/IEC 18033-6:2019, *IT Security techniques – Encryption algorithms – Part 6: Homomorphic encryption*. <https://www.iso.org/standard/67740.html>
[b-ISO/IEC 20889]	ISO/IEC 20889:2018(en), *Privacy enhancing data de-indentification terminology and classification of techniques*. <https://www.iso.org/obp/ui/#iso:std:iso-iec:20889:ed-1:v1:en>
[b-Brakerski]	Brakerski, Z. (2012), *Fully homomorphic encryption without modulus switching from classical GapSVP*. Annual Cryptology Conference. Springer, Berlin, Heidelberg. <https://link.springer.com/chapter/10.1007/978-3-642-32009-5_50>
[b-Brakerski, Z]	Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2011), *Fully Homomorphic Encryption without Bootstrapping*. <https://eprint.iacr.org/2011/277>
[b-Chillotti]	Chillotti, I., Gama, N., Georgieva, M., Izabachène, M. (2020), *TFHE : Fast fully homomorphic encryption over the torus*. Journal of Cryptology. Volume 33, Issue 1, pp 34-91. <https://dl.acm.org/doi/10.1007/s00145-019-09319-x>
[b-Cheon]	Cheon, J.H., Kim, A., Kim, M., and Song, Y. (2017), *Homomorphic Encryption for Arithmetic of Approximate Numbers*. ASIACRYPT 2017: Advances in Cryptology – ASIACRYPT 2017, pp 409–437. <https://link.springer.com/chapter/10.1007/978-3-319-70694-8_15>
[b-Concrete]	"Concrete" library. Github. <https://github.com/zama-ai/concrete>
[b-Ducas]	Ducas, L., and Micciancio,. D. (2015), *FHEW: Bootstrapping homomorphic encryption in less than a second*. Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg. <https://link.springer.com/chapter/10.1007/978-3-662-46800-5_24>
[b-ElGamal]	ElGamal, T. (1985), *A public key cryptosystem and a signature scheme based on discrete logarithms*. IEEE transactions on information theory, Volume: 31, Issue: 4, pp 469-472. <https://ieeexplore.ieee.org/document/1057074/authors#authors>
[b-FCA]	Financial Conduct Authority (2019), *Global AML and Financial Crime TechSprint*. <https://www.fca.org.uk/events/techsprints/2019-global-aml-and-financial-crime-techsprint>
[b-FIPS 186-5]	FIPS 186-5 (2023), *Digital Signature Standard (DSS)*. <https://csrc.nist.gov/pubs/fips/186-5/final>
[b-FIPS 197]	FIPS 197 (2023), *Advanced Encryption Standard*. <https://csrc.nist.gov/pubs/fips/197/final>
[b-FHEW]	Iducas / FHEW library. <https://github.com/lducas/FHEW>

[b-FV12/BFV]	J. Fan and V. Frederik (2012), *Somewhat Practical Fully Homomorphic Encryption*. IACR Cryptology ePrint Archive2012: 144.
[b-FV-NFL]	CryptoExperts / FV-NFLlib library. <https://github.com/CryptoExperts/FV-NFLlib>
[b-Gentry]	Gentry, C. (2009), *Fully homomorphic encryption using ideal lattices*. Proceedings of the forty-first annual ACM symposium on Theory of computing. <https://dl.acm.org/doi/10.1145/1536414.1536440>
[b-HEAAN]	HEaaN Private AI Homomorphic Encryption library. <https://heaan.it/>
[b-HElib]	shaih / HElib library. <https://github.com/shaih/HElib>
[b-iDash]	iDash privacy & security workshop 2024 -secure genome analysis competition. <http://www.humangenomeprivacy.org/>
[b-Lattigo]	Lattigo: lattice-based multiparty homomorphic encryption library in Go library. <https://github.com/ldsec/lattigo>
[b-Narayanan]	Narayanan, A., and Shmatikov, V. (2006), *How to break anonymity of the Netflix prize dataset*. <https://arxiv.org/abs/cs/0610105>
[b-PALISADE]	PALISADE Lattice Cryptography library. <https://gitlab.com/palisade/palisade-release>
[b-Paillier]	Paillier, P. (1999), *Public-key cryptosystems based on composite degree residuosity classes*. EUROCRYPT 1999: Advances in Cryptology – EUROCRYPT '99 pp 223-238. <https://link.springer.com/chapter/10.1007/3-540-48910-X_16>
[b-PKCS]	RFC 8017 (2016), *PKCS #1: RSA Cryptography Specifications Version 2.2*. <https://datatracker.ietf.org/doc/html/rfc8017>
[b-SEAL]	Microsoft / SEAL library. <https://github.com/microsoft/SEAL>
[b-UNH]	United Nations, *UN Handbook on Privacy-Preserving Computation Techniques*. <https://unstats.un.org/bigdata/task-teams/privacy/UN%20Handbook%20for%20Privacy-Preserving%20Techniques.pdf>
[b-WEF]	World Economic Forum (2019), *The Next Generation of Data-Sharing in Financial Services: Using Privacy Enhancing Techniques to Unlock New Value*. White papers. <https://www.weforum.org/publications/the-next-generation-of-data-sharing-in-financial-services-using-privacy-enhancing-techniques-to-unlock-new-value/>

____