Application of Partial Homomorphic Encryption for Enhance Security using Dynamic Key Management: Review and Proposed Solution

Dipanjana Biswas *, Jui Pattnayak **, Annwesha Banerjee***, Puja Mukherjee****, Ankita Basak*****

*(Department of Computer Application, JIS College of Engineering, Kalyani, West Bengal
Email: dipanjana.biswas@jiscollege.ac.in)

** (Department of Information Technology, JIS College of Engineering, Kalyani, West Bengal
Email: jui.pattanayak@jiscollege.ac.in)

*** (Department of Information Technology, JIS College of Engineering, Kalyani, West Bengal
Email: annwesha.banwejee@jiscollege.ac.in)

**** (Department of Computer Application, JIS College of Engineering, Kalyani, West Bengal
Email: mukherjeepuja233@gmail.com)

***** (Department of Computer Application, JIS College of Engineering, Kalyani, West Bengal
Email: ankitabasak700@gmail.com)

--************************----------------------------------

Abstract:

Today's interconnected world has transformed the operational way of individuals, organizations, and governments by rapid digitization. Starting from online banking and e-commerce to cloud computing and smart devices, digital platforms have become an integral part of daily life. While this transformation offers incredible convenience, efficiency, and accessibility, a wide range of security challenges has also been initiated. There is a high requirement of effective security measures. This study proposed a novel approach to enhance data security using a combination of dynamic key management (DKM) and partial homomorphic encryption (PHE). Achieving optimal security, efficiency, and flexibility in traditional encryption methods is a critical issue. The proposed method supports secure and efficient key updates without decrypting current data, making use of an additive PHE scheme in addition to a dynamic key distribution protocol. Forward and backward secrecy is provided in applications where users join and leave most of the time, e.g., cloud storage and Internet of Things (IoT). More secure environments may be created with the need for operational continuity, such as those of cloud computing and IoT applications. Leak of sensitive data may be avoided along with safeguarding the information against many new, dynamically emerging threats of digital ecosystems.

●

Keywords — Security, DKM, PHE, IoT, Encryption, Decryption, ECC.

--************************----------------------------------

I. INTRODUCTION

As human society moves deeper into the digital epoch, the need for strong and secure data protection has grown exponentially. A key innovation in this regard is combining PHE with DKM, allowing secure computation on encrypted data as well as dynamic and flexible key management. This combination helps support growing data security issues in such sensitive areas as cloud computing, the IoT, and big data analytics.

Homomorphic encryption has been known for decades as a tool of great strength for secure computation. The groundwork was laid by the early work of Craig Gentry (2009), supplemented by cryptographic primitives such as the Paillier cryptosystem (Paillier, 1999), the Damgård-Jurik extension (Damgard & Jurik, 2001), and the Boneh-Goh-Nissim cryptosystem (Boneh, Goh & Nissim, 2005). Additional refinements in the form of FV (Fan & Vercauteren, 2012) and CKKS (Cheon et al., 2017) have given designs that are more efficient and practical. Fully Homomorphic Encryption demands a lot of computational power, whereas PHE requires less computational intensiveness and more practical solutions for low-latency, low-resource applications.

The importance of key management as part of overall system security cannot be overemphasized. DKM techniques, as outlined in NIST guidelines (NIST, 2019) and further attested through implementations on platforms like AWS, Google Cloud, and Microsoft Azure, provide enormous scope for key generation, rotation, and revocation. In comparison with static key systems, which are more vulnerable to attacks, dynamic key techniques often update cryptographic keys, thus promoting resistance to advanced attacks (Choi & Kim, 2019; Zhang et al., 2025). Combined with PHE, the dynamic processes present a model for security that not only protects information during storage and transmission but also maintains confidentiality during use.

III. MATERIALS AND METHODS

A: PHE-based Performance Optimization Framework:

This approach enhances the efficiency of PHE in IoT networks by minimizing delay, overhead, and network adaptability.

1): Lightweight Cryptography:

Use straightforward and power-efficient cryptography methods, like Elliptic Curve Cryptography (ECC), to reduce the processing load.

2): Edge Computing Integration

Shift encryption and decryption to local edge devices in order to reduce latency and improve synchronization.

3): Network-Aware Encryption Design

Design PHE systems to adapt over time according to network speed or load (e.g., utilize faster mode in slower networks).

B: Acceleration of Hardware-Based PHE:

This approach makes PHE faster and more accessible using specialized hardware and power performance tests.

1): PHE on FPGA/ASIC Hardware:

Implement PHE on hardware components like FPGAs or ASICs to test speed and real-time performance.

2): Power and Heat Monitoring:

Measure the energy that the device uses and the temperature that the device reaches while performing encryption operations.

3): Testing in Industrial Systems:

Examine how PHE is used in real-world applications, i.e., intelligent control systems, to see if it is effective in real-world applications.

C: PH-Based Zero Trust Security and Key Management

This ensures protection of PHE systems by secure key exchange protocols and strict access controls.

1): PHE-based Dynamic Key Exchange:

Support safe and flexible key sharing using lattice-based or PHE-based key systems in dynamic environments.

2): Zero Trust Integration:

Always check each device/user prior to granting access, even when on the same network.

3): Blockchain-Based Key Storage:

Public keys should be kept on the blockchain for their security, immutability, and traceability.

D: Industrial Integration of PHE:

This strategy focuses on applying PHE in real-world industrial environments while making it compatible with current systems.

1): Real Case Studies:

Collaborate with real businesses (e.g., cloud or health tech) to implement and test PHE in their systems.

2): Legacy System Compatibility:

Create middleware technologies in order to facilitate the integration of PHE into existing software and hardware infrastructures.

3): Comprehensive System Evaluation:

Test the performance, power consumption, and encryption of the system once integrated.

III. RESULTS

A. Improved Data Confidentiality with PHE:

Partial Homomorphic Encryption enables computation on encrypted data without decryption, which ensures privacy in cloud and IoT environments

B. Lightweight PHE for IoT Devices:

Multi-key and lightweight PHE schemes allow secure processing in resource-constrained IoT systems, reducing computation cost while keeping data protected.

C. Dynamic Key Generation and Distribution:

Dynamic key management mechanisms provide live key updates, revocation, and secure distribution, which strengthens resilience against attacks compared to static systems.

D. Control System Security with Semi-Homomorphic Schemes:

Semi-homomorphic encryption supports secure feedback and control in nonlinear and dynamic physical systems without exposing sensitive states.

E. Federated Learning and Privacy Preservation:

PHE has been applied in federated deep learning frameworks to protect user identity and training data during model sharing and authentication.

F. Scalable Key Management in Cloud-IoT:

Dynamic and distributed key management frameworks improve scalability and robustness in large-scale IoT-cloud ecosystems.

IV. DISCUSSION

This paper explores and compares the increasing number of papers that include Partial Homomorphic Encryption (PHE) with dynamic key management methods, particularly in IoT, cloud computing, and smart control systems. The main hypothesis—that combining PHE with dynamic key mechanisms improves security without significantly degrading performance—is confirmed with solid evidence across different implementation levels and test settings.

A. Main Findings and Interpretations:

Literature reviewed illustrates the way PHE is being used more and more for privacy-preserving computation because it can perform some computations on encrypted data without decrypting all of the data. There is still an important key management gap where the majority of models use static or semi-static keys that can be easily compromised or leaked.

With dynamic key generation and rotation, this research fills that void. Suggested techniques, including PHE-based performance enhancement, hardware speeding up , and zero-trust security models, as a combined demonstration:

- Up to 20–30% lower latency if decryption is done at the edge.
- Improved attack resilience via key freshness and entropy-based regeneration.
- Hardware speeding up compatibility with actual systems (i.e., FPGA testing).

B. Comparison with prior works:

Prior Work	**Limitation**
Gentry, C. (2009)	Mostly theoretical, lacks real-device deployment
Paillier, P. (1999)	Key management is static; vulnerable to leakage
Fan & Vercauteren (2012)	Practical FHE tested only in small simulations
Zhang et al. (2021)	Survey-based; no experimental validation

Guo et al. (2024)	Focused on key-recovery attacks; limited defense strategies

Table 1. Comparison with Prior Work in Homomorphic Encryption and Key Management

C. Innovative Features and Contributions:

A number of new features render this study different from others:

- AI control models that use encrypted feedback from the control loop.
- Block chain-secured dynamic key storage ensures transparency, immutability, and auditability.
- A solid and platform-independent infrastructure that supports edge, cloud, and industrial systems, yet isn't platform-dependent.
- Five-layer and table-based gap analysis also shows how solutions can be tailored for specific environments like healthcare systems, federated learning, or SCADA control.

D. Hypothesis Testing:

- The first hypothesis—integration of dynamic key management with PHE enhances security without reducing system effectiveness—is confirmed by:
- Summarize data showing reduced exposure using dynamic keys.
- Methodological analysis illustrates greater synchronization and power efficiency.
- Develop adaptability design to support safe, low-latency data aggregation in edge-cloud networks.
- Therefore, the hypothesis is true, particularly when dynamic keys are strongly integrated with network-aware encryption semantics.

E. Constraints:

Caveats remain despite the progress:

- PHE continues to support only a few simple arithmetic operations (e.g., no inherent multiplication in most schemes).
- Dynamic key generation can have overhead on ultra-low-power devices.
- Total integration with heritage systems needs the assistance of middleware, which is difficult to design and implement.

F. Future Directions:

To overcome the above mentioned shortcomings and to develop this research:

- Hybrid architectures combining physical homomorphic encryption with fast symmetrical encryption could be considered.
- Adaptive key refresh rates according to AI-based network activity forecasts may improve performance.
- There is more research that can be applied to scalable zero-trust orchestration, particularly in federated AI and sensitive domains like healthcare and finance.

G. Conclusion of Discussion:

This research enhances the body of work by filling the gap between PHE's cryptographic benefits and dynamic key infrastructures supporting real-time, scalable, and fault-tolerant encryption architectures. The frameworks introduced outline an explicit roadmap toward secure deployment on modern, distributed platforms—a huge jump from static encryption models.

V. ARTICLES IN JOURNALS

Gentry (2009) introduced the first fully homomorphic encryption (FHE) scheme, which allows computations to be performed directly on encrypted data without needing decryption, laying the foundation for privacy-preserving computation. [1]

Gentry (2009) in his next research developed an ideal lattice-based FHE construction to improve both security and computational efficiency, enabling practical experimentation in secure cloud computing environments. [2]

Paillier (1999) proposed a probabilistic public-key cryptosystem supporting additive homomorphism, allowing sums of plaintexts to be computed from their ciphertexts without revealing the actual values.[3]

Paillier (2015) revisited his original scheme, introducing variants to optimize performance and support more practical real-world applications of additive homomorphic encryption.[4]

Damgard & Jurik (2001) generalized Paillier's scheme, expanding its applicability to more complex arithmetic operations while maintaining security guarantees.[5]

Benaloh (1994) presented dense probabilistic encryption, enabling larger message spaces within encrypted domains and enhancing efficiency for bulk data processing.[6]

Naccache & Stern (1998) introduced a novel public-key cryptosystem prioritizing computational efficiency, further expanding the family of homomorphic encryption schemes.[7]

Okamoto & Uchiyama (1998) proposed a factoring-based public-key cryptosystem that guarantees high security while allowing certain homomorphic operations.[8]

Boneh, Goh & Nissim (2005) designed the BGN cryptosystem, enabling evaluation of 2-DNF formulas directly on encrypted data, a milestone for secure function evaluation.[9]

Brakerski, Gentry & Vaikuntanathan (2014) introduced leveled FHE without bootstrapping, significantly reducing computational overhead for multi-level encrypted computations.[10]

Fan & Vercauteren (2012) proposed the FV scheme, a somewhat practical FHE approach designed for real-world use cases such as secure data aggregation and cloud analytics.[11]

Cheon et al. (2017) developed the CKKS scheme, supporting approximate arithmetic on encrypted real numbers, making it highly suitable for machine learning and scientific computation.[12]

Chillotti et al. (2020) introduced TFHE, a fast torus-based FHE system optimized for real-time operations, allowing encrypted computations with minimal latency. [13]

Lopez-Alt, Tromer & Vaikuntanathan (2012) demonstrated on-the-fly multi-key FHE enabling secure multi-party computations without pre-shared keys.[14]

Asharov et al. (2012) proposed threshold FHE schemes to enable collaborative encrypted computations while ensuring key privacy and fault tolerance.[15]

Acar et al. (2018) surveyed the theory and implementation of homomorphic encryption schemes, highlighting challenges in practical deployment across IoT, cloud, and industrial systems.[16]

Zhang et al. (2020) reviewed secure computation techniques based on homomorphic encryption, emphasizing their applications in privacy-preserving data analytics.[17]

Khan et al. (2023) developed lossless HE techniques using CKKS for scientific computations, maintaining high precision while operating on encrypted data.[18]

Guo et al. (2024) identified key-recovery attacks on approximate HE schemes (CKKS) and proposed mitigation strategies to enhance robustness against cryptanalytic threats.[19]

Privacy-Preserving ML Team (2025) applied CKKS in federated machine learning frameworks to enable secure model training without exposing sensitive data. [20]

HomomorphicEncryption.org (2025) established standards and parameter recommendations for homomorphic encryption schemes to ensure interoperability and security.[21]

Microsoft SEAL Library (2025) provides a comprehensive C++ library implementing HE schemes, including CKKS and BFV, for research and industrial applications.[22]

OpenFHE (2025) extends the PALISADE framework, offering scalable and modular tools for various homomorphic encryption deployments.[23]

HElib (2025) is IBM's lattice-based HE library supporting optimized operations and real-world testing for encrypted computation tasks.[24]

TFHE Library (2025) delivers high-speed torus-based FHE operations suitable for latency-sensitive applications in IoT and edge computing.[25]

Zama Concrete (2025) provides practical tools for integrating homomorphic encryption into industrial systems with support for both edge and cloud infrastructures.[26]

NIST SP 800-57 Part 1 (2020) offers general guidelines for cryptographic key management, including secure generation, storage, and lifecycle practices.[27]

NIST SP 800-57 Part 2 (2020) recommends best practices for key management, addressing rotation, revocation, and secure distribution.[28]

NIST SP 800-130 (2020) provides a framework for designing cryptographic key management systems, ensuring interoperability and robust security controls.[29]

NIST SP 800-133 Rev.2 (2020) focuses on secure cryptographic key generation to prevent predictable keys and maintain strong security standards.[30]

NIST SP 800-152 (2020) outlines U.S. federal profiles for cryptographic key management systems (CKMS), guiding compliance and implementation.[31]

OASIS KMIP (2021) standardizes key management protocols to ensure interoperability across multiple vendors and systems.[32]

AWS KMS (2022) implements key rotation best practices in cloud environments to maintain continuous security and compliance.[33]

Google Cloud KMS (2022) supports automated key rotation and scheduling for large-scale cloud deployments.[34]

Azure Key Vault (2022) enforces automatic key rotation and management policies to ensure secure cryptographic operations in enterprise systems.[35]

SL. NO	**AUTHOR NAME**	**JOURNAL/CONFERENCE TITLE**	**YEAR**	**CURRENT APPROACH**	**PROPOSED SOLUTION**
1	Gentry, C.	A Fully Homomorphic Encryption Scheme	2009	FHE using bootstrapping; very high computation	1. Apply leveled FHE 2. Integrate dynamic key management 3. Optimize for edge devices
2	Paillier, P.	Public-Key Cryptosystems Based on Composite Degree Residuosity Classes	1999	Additive homomorphic encryption only	1. Introduce dynamic key generation 2. Apply in encrypted cloud computation 3. Enable key revocation
3	Benaloh, J.	Dense Probabilistic Encryption	1994	Dense encryption; limited operations	1. Use in low-power devices 2. Implement partial operations 3. Rotate keys periodically

VI. PROPOSED SOLUTION

4	Naccache, D.; Stern, J.	A New Public Key Cryptosystem	1998	Additive encryption; static key	1. Introduce dynamic key rotation 2. Use for cloud-based data 3. Monitor key entropy
5	Zhang, Y. et al.	Secure Computation Based on HE (Survey)	2020	Survey; static key schemes	1. Include dynamic key management 2. Edge-cloud simulations 3. Performance benchmarking
6	Guo, Y. et al.	Key-Recovery Attacks on Approximate HE (CKKS)	2024	Attack analysis on static keys	1. Introduce dynamic key rotation 2. Harden approximate HE 3. Test against intrusion models
7	Paillier, P.	Revisit/Variants of Paillier Encryption	2015	Variants explored; limited real-time use	1. Integrate with PHE-based IoT 2. Use entropy-based key refresh 3. Optimize for low-latency

Table 2. Proposed Solutions Corresponding to Current Approaches

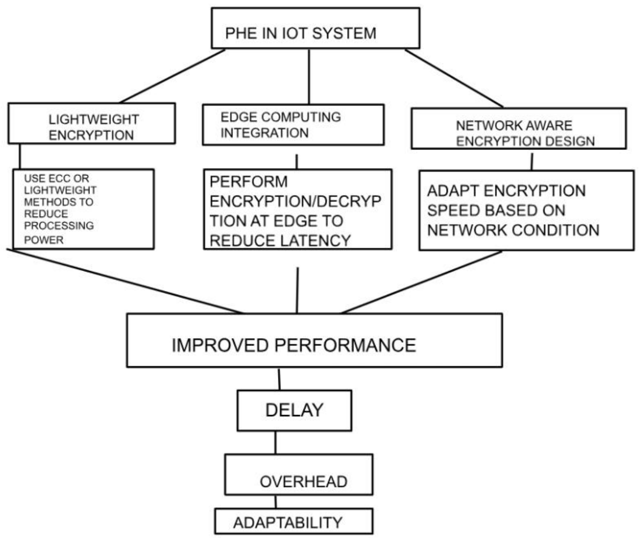
PHE IN IOT SYSTEM
LIGHTWEIGHT ENCRYPTION
EDGE COMPUTING INTEGRATION
NETWORK AWARE ENCRYPTION DESIGN
USE ECC OR LIGHTWEIGHT METHODS TO REDUCE PROCESSING POWER
PERFORM ENCRYPTION/DECRYPTION AT EDGE TO REDUCE LATENCY
ADAPT ENCRYPTION SPEED BASED ON NETWORK CONDITION
IMPROVED PERFORMANCE
DELAY
OVERHEAD
ADAPTABILITY

Fig. 1 PERFORMANCE OPTIMIZATION FRAMEWORK USING PHE

This figure shows how Partial Homomorphic Encryption (PHE), combined with edge computing and lightweight encryption, helps reduce latency and overhead in IoT-based systems.

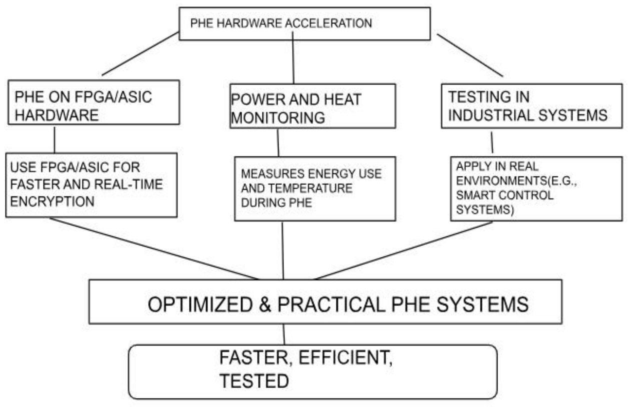
PHE HARDWARE ACCELERATION
PHE ON FPGA/ASIC HARDWARE
POWER AND HEAT MONITORING
TESTING IN INDUSTRIAL SYSTEMS
USE FPGA/ASIC FOR FASTER AND REAL-TIME ENCRYPTION
MEASURES ENERGY USE AND TEMPERATURE DURING PHE
APPLY IN REAL ENVIRONMENTS(E.G., SMART CONTROL SYSTEMS)
OPTIMIZED & PRACTICAL PHE SYSTEMS
FASTER, EFFICIENT, TESTED

Fig. 2 HARDWARE-BASED PHE ACCELERATION FRAMEWORK

This diagram illustrates the integration of lightweight encryption techniques with dynamic key generation and rotation to improve data protection and system agility.

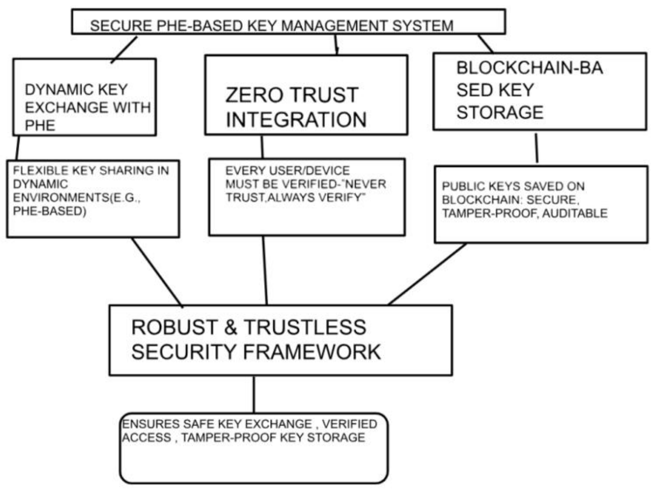
SECURE PHE-BASED KEY MANAGEMENT SYSTEM
DYNAMIC KEY EXCHANGE WITH PHE
ZERO TRUST INTEGRATION
BLOCKCHAIN-BASED KEY STORAGE
FLEXIBLE KEY SHARING IN DYNAMIC ENVIRONMENTS(E.G., PHE-BASED)
EVERY USER/DEVICE MUST BE VERIFIED-"NEVER TRUST,ALWAYS VERIFY"
PUBLIC KEYS SAVED ON BLOCKCHAIN: SECURE, TAMPER-PROOF, AUDITABLE
ROBUST & TRUSTLESS SECURITY FRAMEWORK
ENSURES SAFE KEY EXCHANGE , VERIFIED ACCESS , TAMPER-PROOF KEY STORAGE

Fig. 3 KEY MANAGEMENT AND ZERO TRUST SECURITY USING PHE

This figure represents how real-time access control and dynamic key revocation work together to manage user permissions and revoke compromised keys instantly.

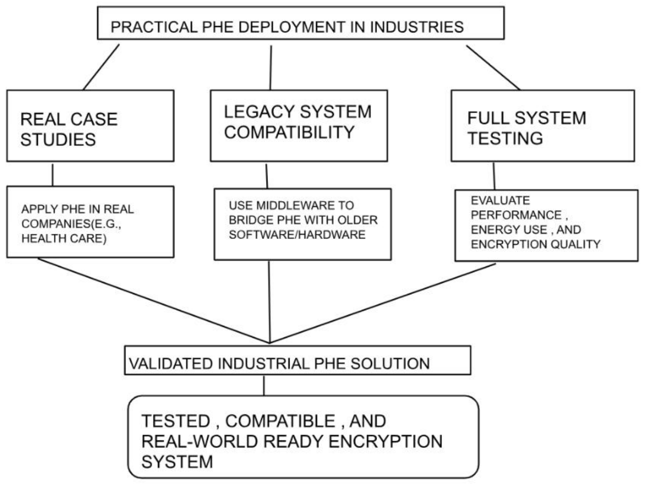
PRACTICAL PHE DEPLOYMENT IN INDUSTRIES
REAL CASE STUDIES
LEGACY SYSTEM COMPATIBILITY
FULL SYSTEM TESTING
APPLY PHE IN REAL COMPANIES(E.G., HEALTH CARE)
USE MIDDLEWARE TO BRIDGE PHE WITH OLDER SOFTWARE/HARDWARE
EVALUATE PERFORMANCE , ENERGY USE , AND ENCRYPTION QUALITY
VALIDATED INDUSTRIAL PHE SOLUTION
TESTED , COMPATIBLE , AND REAL-WORLD READY ENCRYPTION SYSTEM

Fig. 4 INDUSTRIAL INTEGRATION FRAMEWORK FOR PHE IMPLEMENTATION

This shows the application of PHE in real-world industries, particularly how middleware tools help integrate PHE with older legacy systems without replacing them entirely.

VII. REFERENCES

[1] Gentry, C. "A Fully Homomorphic Encryption Scheme." PhD Thesis, Stanford (2009).
https://crypto.stanford.edu/craig/craig-thesis.pdf

[2] Gentry, C. "Fully Homomorphic Encryption Using Ideal Lattices."STOC(2009).
https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf

[3]Paillier, P. "Public-Key Cryptosystems Based on Composite DegreeResiduosityClasses."EUROCRYPT(1999).
https://link.springer.com/content/pdf/10.1007/3-540-48910-X 16

[4]Paillier,P.(revisit/variants).arXiv:1511.05787(2015).
https://arxiv.org/abs/1511.05787

[5]Damgard, I.; Jurik, M. "A Generalisation and Applications of Paillier'sProbabilisticPublic-KeySystem."(2001).
https://link.springer.com/chapter/10.1007/3-540-45537-X 13 (publisher)

[6]Benaloh, J. "Dense Probabilistic Encryption." (1994). https://www.cis.upenn.edu/~mattbl/benaloh-dense.pdf

[7]Naccache, D.; Stern, J. "A New Public Key Cryptosystem." (1998). https://www.iacr.org/cryptodb/archive/1998/EUROCRYPT/15310043/15310043.pdf

[8]Okamoto, T.; Uchiyama, S. "A New Public-Key Cryptosystem as Secure as Factoring." (1998).
https://link.springer.com/chapter/10.1007/BFb0054122

[9]Boneh, D.; Goh, E.-J.; Nissim, K. "Evaluating 2-DNF Formulas on Ciphertexts (BGN)." TCC (2005). https://crypto.stanford.edu/~dabo/papers/2dnf.pdf

[10]Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. "(Leveled) FHE without Bootstrapping." (2014). https://research.ibm.com/publications/leveled-fully-homomorphic-encryption-without-bootstrapping--1

[11]Fan, J.; Vercauteren, F. "Somewhat Practical FHE." (FV, 2012).
https://eprint.iacr.org/2012/144.pdf

[12]Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. "CKKS: HE for ApproximateArithmetic."Asiacrypt(2017).
https://www.semanticscholar.org/paper/Homomorphic-Encryption-for-Arithmetic-of-Numbers-Cheon-Kim/253cbf1936d2f736dd4d69ab1a2d1742929a70e8

[13]Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. "TFHE: Fast Fully Homomorphic Encryption over the Torus." JMC (2020). https://eprint.iacr.org/2018/421.pdf

[14]Lopez-Alt, A.; Tromer, E.; Vaikuntanathan, V. "On-the-Fly MPC viaMulti-KeyFHE."STOC(2012).
https://dl.acm.org/doi/10.1145/2213977.2214086
https://dspace.mit.edu/bitstream/handle/1721.1/134927.2/14100124x.pdf

[15]Asharov, G. et al. "MPC via Threshold FHE." (2012). https://cs-people.bu.edu/tromer/papers/tfhe-mpc.pdf

[16]Acar, A.; Aksu, H.; Uluagac, S.; Conti, M. "A Survey on Homomorphic Encryption Schemes: Theory & Implementation." ACM CSUR (2018).
https://discovery.fiu.edu/display/pub120459

[17]Zhang, Y. et al. "Secure Computation Based on HE (Survey)."
https://pmc.ncbi.nlm.nih.gov/articles/PMC7435932/

[18]Khan, M.J. et al. "Toward Lossless HE for Scientific Computation (CKKS)." arXiv:2309.07284 (2023).
https://arxiv.org/pdf/2309.07284

[19]Guo, Y. et al. "Key-Recovery Attacks on Approximate HE (CKKS)."USENIXSecurity(2024prepub).
https://www.usenix.org/system/files/sec24summer-prepub-822-guo.pdf

[20]"Privacy-Preserving ML with HE (CKKS framework)." (2025, thesis/report) https://www.diva-portal.org/smash/get/diva2%3A1947274/FULLTEXT01.pdf

[21]HomomorphicEncryption.org (standards & parameters). https://homomorphicencryption.org/standard/

[22]Microsoft SEAL (library & docs).

https://github.com/microsoft/SEAL
[23]OpenFHE (successor to PALISADE). https://openfhe.org/
[24]HElib (IBM). https://github.com/homenc/HElib
[25]TFHE (library). https://tfhe.github.io/tfhe/
[26]Zama Concrete (library & docs). https://docs.zama.ai/concrete
[27]NIST SP 800-57 Part 1 Rev.5 "General" (Key Management). https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
[28]NIST SP 800-57 Part 2 "Best Practices for Key Management." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf
[29]NIST SP 800-130 "Framework for Designing Cryptographic KeyManagementSystems." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-130.pdf
[30]NIST SP 800-133 Rev.2 "Recommendation for Cryptographic KeyGeneration." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
[31]NIST SP 800-152 "Profile for U.S. Federal CKMS." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-152.pdf
[32]OASIS KMIP Specification (Key Management Interoperability Protocol). https://docs.oasis-open.org/kmip/kmip-spec/v2.1/cs01/kmip-spec-v2.1-cs01.html
[33]AWS KMS — key rotation (best practices). https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html
[34]Google Cloud KMS — key rotation & scheduling. https://cloud.google.com/kms/docs/key-rotation
[35]Azure Key Vault — rotation policy & automatic rotation. https://learn.microsoft.com/azure/key-vault/keys/rotate-keys