An Expert-Level Analysis of Homomorphic Encryption: Mathematical Principles, Applications, and Computational Challenges

1. Introduction to Homomorphic Encryption: The Cryptographic Holy Grail

1.1 What is Homomorphic Encryption?

Homomorphic Encryption (HE) is a highly advanced cryptographic technique that enables computation on encrypted data, or ciphertext, without requiring it to be decrypted first.[1] The fundamental purpose of HE is to ensure that data remains confidential while it is being processed, thereby enabling useful and complex tasks to be accomplished in untrusted environments, such as cloud computing.[3] Unlike traditional encryption, which secures data "at rest" (in storage) and "in transit" (during transfer), HE provides a critical layer of protection for "data in use".[5] This capability is a significant departure from conventional cryptographic methods, which necessitate data decryption for any form of processing, thereby creating a moment of vulnerability.[2]

A homomorphic cryptosystem functions similarly to other forms of public-key encryption in its use of a public key for encryption and a corresponding private key for decryption.[3] Its distinctiveness, however, lies in its use of an underlying algebraic system that permits operations on the encrypted data.[3] When these operations are performed on ciphertexts, the result is a new ciphertext which, upon decryption with the private key, yields the same result as if the operations had been performed on the original, unencrypted plaintext.[4] This paradigm shift means that data can be outsourced to a third-party processor, such as a cloud

service provider, for analysis or computation while the sensitive information remains fully protected.[4] This effectively resolves the historical conflict between data utility and data confidentiality, enabling new opportunities for collaboration and value creation.

1.2 The Homomorphic Property: A Mathematical Analogy

The core principle of homomorphic encryption is rooted in the concept of a homomorphism in algebra.[4] This mathematical property establishes a relationship where an operation in one algebraic system (e.g., addition in the ciphertext space) corresponds directly to an operation in another algebraic system (e.g., addition in the plaintext space).[5] This relationship can be formally expressed by the equation:

$$Dec(f'(Enc(x)))=Dec(Enc(f(x)))$$ [5]

Here, Enc(x) represents the encrypted form of the plaintext x, and Dec is the decryption function. The function f′ is the homomorphic version of the plaintext function f. The equation formalizes the fact that performing a homomorphic operation on a ciphertext and then decrypting it is equivalent to first decrypting the plaintext and then performing the operation on it.[5] This fundamental property is what enables computation on encrypted data.

A more intuitive way to understand this is through a conceptual analogy. Traditional encryption is often compared to a securely locked box containing a valuable message.[5] To access or manipulate the contents, one must possess the secret key to unlock the box, which inevitably creates a moment of vulnerability.[2] In this model, the data is either secured (inside the locked box) or accessible (outside the box), but never both simultaneously.

Homomorphic encryption, however, is a more sophisticated design.[5] The locked box still protects the data, but it is equipped with a pair of "special built-in gloves".[5] These gloves allow a third-party processor to reach inside the box and manipulate the contents—for example, by adding or multiplying values—without ever having to unlock it or "see" the data within.[2] Only the owner with the private key can unlock the box at the end of the process and view the final, manipulated result.[5] This powerful visual representation underscores how HE maintains confidentiality even when data is actively being used and processed, distinguishing it as a significant leap forward in data security.

2. A Taxonomy of Homomorphic Encryption

Homomorphic encryption is not a single technology but a family of schemes with varying capabilities and limitations. The categorization of these schemes—Partially, Somewhat, and Fully Homomorphic Encryption—provides a clear understanding of their respective trade-offs between functionality, performance, and operational capacity. The progression through these types reflects the evolution of the field, with each new type addressing the limitations of its predecessor.

2.1 Partially Homomorphic Encryption (PHE)

Partially Homomorphic Encryption is the most basic and, in many cases, the most computationally efficient form of HE.[8] Schemes in this category support only a single type of operation, either addition or multiplication, but allow that single operation to be performed an unlimited number of times.[3] This means that while they are limited in versatility, they can handle specific, repetitive tasks with high efficiency.[8]

Prominent examples include the Paillier cryptosystem, which is additively homomorphic, and the Rivest-Shamir-Adleman (RSA) cryptosystem, which is multiplicatively homomorphic.[3] Paillier's additive property, for instance, makes it a suitable choice for secure voting applications, as it allows for the unbiased summation of votes while keeping individual ballots confidential.[3] The primary limitation of PHE is its inability to support a combination of operations, which severely restricts its utility for more complex computations.[11]

2.2 Somewhat Homomorphic Encryption (SHE)

Somewhat Homomorphic Encryption schemes represent a step up in functionality. They are capable of performing multiple types of operations—typically both addition and multiplication—but only up to a fixed, predetermined number of times.[3] This limitation is a direct consequence of a fundamental property of most HE schemes: the presence of "noise".[1]

Noise is a small, random term intentionally added to the ciphertext during encryption to ensure its security.[5] With each successive homomorphic operation, the level of noise in the ciphertext grows.[1] If this noise accumulates beyond a certain threshold, it can corrupt the data and lead to decryption failure or an incorrect result.[1] This growth in noise effectively imposes a "bounded depth" on the number of computations that can be performed, which is

why SHE schemes are only suitable for scenarios involving a limited number of operations.[1] Examples of SHE schemes include Yao's Garbled Circuits and the Boneh-Goh-Nissim (BGN) scheme.[10]

2.3 Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption is considered the "holy grail" of cryptography because it enables an unlimited, unbounded number of both addition and multiplication operations on encrypted data.[3] This powerful capability makes FHE schemes Turing-complete, meaning they can perform any computable function.[11] The breakthrough that distinguishes FHE from SHE is the ability to manage the noise accumulation problem, which is the very limitation that plagues SHE schemes.[9]

The key to achieving this unbounded functionality is a process known as **bootstrapping**.[1] Bootstrapping allows for the "refreshing" of a ciphertext by homomorphically evaluating the decryption function on it.[1] This process effectively reduces the noise level to a manageable state, creating a "fresh" ciphertext that can undergo further computation without risk of corruption.[1]

The evolution from SHE to FHE demonstrates a clear progression in addressing a core technical limitation. The inherent problem of noise in SHE directly necessitated the invention of a complex solution like bootstrapping to enable FHE. This cause-and-effect relationship defines the field's trajectory and highlights how modern research has transformed a theoretical concept into a viable, albeit complex, technology.[9]

The following table provides a succinct overview of the key differences among the three main types of homomorphic encryption.

Scheme Type	Supported Operations	Operational Depth	Computational Efficiency	Primary Limitation	Example Schemes
Partially (PHE)	One type (addition or multiplication)	Unlimited	Fastest	Limited versatility	RSA, Paillier
Somewhat	Limited	Bounded	Moderate	Noise	BGN,

(SHE)	number of both			accumulation	YASHE
Fully (FHE)	Unlimited number of both	Unbounded	Most intensive	High computational overhead	Gentry, BGV, BFV, CKKS

3. The Mathematical and Algorithmic Core of FHE

The development of modern homomorphic encryption schemes is built upon a foundation of complex mathematics and sophisticated algorithms. These schemes are designed to balance the competing demands of security, functionality, and performance.

3.1 The Learning with Errors (LWE) Problem

The security of the majority of modern FHE schemes, including the prominent BGV, BFV, and CKKS schemes, is based on the presumed hardness of the Learning with Errors (LWE) problem or its variant, Ring-LWE (RLWE).[4] LWE is a mathematical problem in lattice-based cryptography that involves finding a secret key from a set of linear equations where a small, random "error" term has been added.[5] The problem is believed to be computationally difficult to solve with both classical and quantum computers, which provides FHE with a strong security foundation, making it quantum-resistant.[15] This dependence on lattice-based problems differentiates them from older cryptosystems like RSA, which are considered vulnerable to quantum attacks.[16]

3.2 The Engine of FHE: Noise and Bootstrapping

As discussed in the taxonomy, noise is a critical component of HE. It is a necessary evil—a random term added during encryption that ensures security, but one that grows with each operation and can lead to decryption failure if left unchecked.[1] The most significant increase

in noise typically occurs during homomorphic multiplication.[14]

The breakthrough that enabled FHE was the invention of **bootstrapping** by Craig Gentry in his seminal 2009 thesis.[1] The bootstrapping process is an ingenious mechanism to "refresh" a ciphertext that has accumulated excessive noise.[1] It works by using the HE scheme itself to perform the decryption function on its own ciphertext.[5] In essence, the scheme encrypts its own secret key and then homomorphically applies a decryption circuit to the noisy ciphertext.[5] The result is a new ciphertext that encrypts the same original plaintext but with a significantly lower level of noise, effectively resetting the noise counter and enabling an unlimited number of subsequent computations.[4] This process, while computationally intensive, is what gives FHE its powerful, unbounded functionality.

3.3 Modern FHE Schemes: A Deeper Look

The evolution of FHE has moved beyond Gentry's foundational proof-of-concept to more efficient and specialized schemes. This progression reflects a move from answering the theoretical question of "Can we build a fully homomorphic scheme?" to the practical engineering question of "How can we build a scheme that is efficient for a specific task?"

3.3.1 BFV and BGV

The BFV (Brakerski/Fan-Vercauteren) and BGV (Brakerski-Gentry-Vaikuntanathan) schemes are both considered second-generation FHE systems.[4] They are both based on the Ring-LWE problem and are designed for exact integer arithmetic.[17] Both schemes exhibit similar noise behavior, with homomorphic multiplication causing a higher rate of noise growth than addition.[17]

A key technical difference lies in their noise management strategies. The BGV scheme is "scale-dependent," which means it utilizes a chain of nested ciphertext moduli. As computations are performed, ciphertexts move from a higher-level modulus to a lower-level one, effectively reducing noise through a process known as modulus switching.[17] In contrast, the BFV scheme is "scale-invariant," maintaining a single ciphertext modulus throughout the homomorphic evaluation.[17] The plaintext message is also handled differently: BFV places the message towards the Most Significant Bits (MSBs) of the ciphertext coefficient, while BGV places it towards the Least Significant Bits (LSBs).[17]

3.3.2 CKKS

The CKKS (Cheon-Kim-Kim-Song) scheme is a major departure from BFV and BGV, as it is designed specifically for approximate arithmetic on real and complex numbers.[19] This makes it exceptionally well-suited for applications where a small degree of precision loss is acceptable, such as in machine learning, data science, and signal processing.[19]

The mathematical foundation of CKKS involves the use of cyclotomic polynomials and canonical embedding.[20] The scheme encodes a vector of real or complex values into a polynomial.[20] This encoding is an isomorphism (a one-to-one homomorphism) that leverages the properties of these mathematical structures to enable efficient homomorphic operations.[20] The process of converting a vector into a polynomial and vice-versa often requires solving a linear equation involving a Vandermonde matrix.[20]

The existence of a scheme like CKKS demonstrates a clear trend in the HE field towards specialization and practical optimization. Instead of a single, all-purpose solution, researchers are creating tailored schemes that address the specific data types and computational requirements of high-value applications, which is a powerful indicator of the technology's maturation.

Scheme Name	Data Type	Noise Management	Plaintext Representation	Primary Application
BFV	Integer (Exact)	Scale-invariant (constant modulus)	Most Significant Bits	General integer arithmetic, comparison operations
BGV	Integer (Exact)	Scale-dependent (modulus switching)	Least Significant Bits	General integer arithmetic, comparison operations
CKKS	Real/Complex	Bootstrapping with	Canonical	Machine learning, data

	(Approximate)	approximate numbers	embedding	analytics

4. Real-World Applications and Use Cases

The ability of homomorphic encryption to protect data while it is in use has profound implications for a wide range of industries that handle sensitive information. HE provides a technological solution to a global regulatory and ethical imperative for data privacy. The following sections detail some of the most compelling current and emerging use cases.

4.1 Cloud Computing

Cloud computing is a prime application for homomorphic encryption.[4] As organizations outsource data storage and processing to commercial cloud environments, they face the risk of data breaches and a loss of control over their sensitive information.[6] With HE, an organization can encrypt its data and upload it to the cloud. The cloud service provider can then perform computations—such as data analytics, statistical analysis, or machine learning—on the encrypted data without ever having access to the unencrypted values.[3] The results are returned to the data owner in an encrypted form, and only the owner can decrypt the final output.[22] This framework allows businesses to leverage the scalable and cost-effective resources of the cloud without compromising data confidentiality.[5]

4.2 Healthcare and Genomic Analysis

The healthcare sector is a major beneficiary of homomorphic encryption due to strict privacy regulations like HIPAA and GDPR.[23] Medical researchers and providers often need to collaborate and analyze large, sensitive datasets, such as patient medical histories and genetic information, to identify disease patterns and develop new treatments.[23] This is often impossible due to the risk of exposing personal health information.

Homomorphic encryption provides a solution by allowing multiple hospitals or research

institutions to securely pool and analyze encrypted patient data without ever needing to decrypt it.[24] This enables complex computations, such as predictive analysis of medical data, and allows for collaborative research that would otherwise be legally and ethically unfeasible.[2] The use of HE in this context not only improves data security but also helps build patient trust, which in turn encourages individuals to contribute their data for scientific purposes.[24] An early example of this application was the Bio-Cryptosystem, developed in 2003, which used the Paillier cryptosystem to protect genetic data.[23]

4.3 Finance and Fraud Detection

The financial services industry deals with sensitive data on a global scale, where cross-border data transfer is complicated by various regulatory requirements.[25] Fraud detection, for example, often requires banks to analyze customer data from different jurisdictions.[22]

Mastercard's pilot program with Singapore's Infocomm Media Development Authority provides a compelling example of HE in action.[25] The system uses FHE to allow a secure query hub to check encrypted International Bank Account Numbers (IBANs) for fraud risk. The system returns a simple "true/false" risk result, while the raw IBAN data remains encrypted and localized, never crossing jurisdictional borders.[25] This approach elegantly satisfies complex data localization regulations while enabling essential, real-time fraud analysis, demonstrating the practical value of HE in reconciling business needs with legal and privacy requirements.[25]

4.4 Secure Voting and Public Accountability

The integrity of democratic processes relies on two seemingly contradictory principles: the privacy of the ballot and the verifiability of the vote count. Homomorphic encryption offers a solution that addresses both.[2]

With an HE-based system, each voter's ballot can be encrypted. The votes can then be tallied homomorphically, revealing only the final count without ever disclosing how any individual voted.[3] Microsoft's ElectionGuard is a notable example of this application. It uses HE to ensure accurate voting results and provides voters with tracking codes that allow them to independently verify that their encrypted vote was correctly counted in the final tally.[22] This enhances the security and transparency of the election process, fostering greater public

confidence.

5. The Grand Challenge: Computational Overhead and Performance

Despite its transformative potential, homomorphic encryption is not yet a mainstream technology. The primary barrier to its widespread adoption is the significant computational overhead associated with its use.[8] The complexities of HE schemes introduce a number of performance and resource challenges that are actively being addressed by the research community.

5.1 Quantifying the Cost

The most notable challenge is the sheer slowness of HE operations compared to their plaintext equivalents. Operations on encrypted data are often thousands of times slower than on unencrypted data.[8] Early implementations of Gentry's foundational FHE scheme were particularly slow, with a reported timing of approximately 30 minutes per basic bit operation.[4] While subsequent optimizations have improved this by "many orders of magnitude," the latency remains a major hurdle for real-time applications and is not yet practical for the average user.[4]

In addition to latency, FHE requires a significant investment in computational resources. It demands "extremely high computational power, storage, and energy" to manage the complex mathematics and algorithms involved.[8] This makes it impractical for resource-constrained environments, such as mobile or edge computing devices, and generally confines its use to datacenter-scale computing.[15]

5.2 The "Holistic" Problem

The challenges of homomorphic encryption extend beyond pure performance metrics. The implementation and management of these systems are notoriously complex, requiring specialized knowledge and expertise in advanced cryptography and algebra.[8] This complexity

creates a barrier to entry for many organizations, limiting its current use to enterprise-level organizations with ample resources.[8]

Furthermore, homomorphic encryption schemes are inherently malleable.[4] This property, which allows the data to be modified while encrypted, is a necessary feature for its functionality. However, it also means that HE schemes have weaker security properties than non-homomorphic schemes, as they can be manipulated.[4] This trade-off must be carefully managed in a real-world implementation. The performance penalty is also not uniform across all schemes. Simpler PHE schemes are the most efficient, while the more versatile FHE schemes are the most computationally intensive.[8] This requires a careful, use-case-specific analysis to determine the optimal trade-off between functionality and efficiency.[8]

These challenges highlight that the high computational cost of FHE is not merely a technical limitation but a multifaceted engineering problem. It is a fundamental barrier that is driving a new wave of research and development focused on creating practical, efficient, and user-friendly HE systems.

6. Pathways to Practicality: Recent Research and Optimizations

The challenges of computational overhead and implementation complexity are not insurmountable. The field of homomorphic encryption is transitioning from a theoretical phase to a more practical, applied engineering phase, with significant research efforts dedicated to making HE a viable technology for a broader range of applications.

6.1 Compiler-Based Optimization

Programming efficient homomorphic encryption applications is widely considered difficult due to the restricted nature of operations and the need for meticulous noise management.[27] To address this, researchers are developing

FHE compilers that automate the process of translating high-level code into secure and efficient HE implementations.[27]

These compilers, such as Google's compiler which converts C++ code into FHE ciphertexts, employ a variety of optimization techniques.[29] For instance, they can transform higher-order

data types (e.g., integers, arrays) into boolean circuits, unroll loops, and use circuit synthesis suites like Yosys to optimize the resulting circuit for size and efficiency.[29] The development of these tools is critical for democratizing HE and making it accessible to developers who lack deep expertise in the underlying cryptography.[28]

6.2 Hardware Acceleration

The high computational demands of FHE schemes, particularly for power-constrained devices, have driven research into hardware acceleration.[15] The Number Theory Transform (NTT), a core component of many lattice-based FHE schemes for performing polynomial multiplication, is a major performance bottleneck.[15]

To address this, researchers are designing specialized hardware accelerators, such as integrating an NTT within a RISC-V architecture.[15] This approach leverages the parallelization capabilities of the NTT and the power efficiency of the RISC-V architecture to significantly speed up the polynomial multiplication process.[15] This engineering effort aims to bring FHE from the domain of datacenter-scale computing to resource-constrained environments like edge devices, thereby expanding its potential applications.[15]

6.3 The Next Frontier: Third and Fourth-Generation Schemes

The evolution of HE schemes has not stalled since the advent of BGV and BFV. The third generation of schemes, such as the Gentry, Sahai, and Waters (GSW) scheme, introduced novel methods for homomorphic operations that avoid key and modulus switching.[30] The FHEW scheme, also from this generation, achieved a bootstrapping time of under one second, which was a major advancement in addressing the performance bottleneck of this process.[30]

The fourth generation of FHE schemes is exemplified by the CKKS scheme and represents a shift toward higher computational efficiency through approximate computation.[13] This focus on approximate arithmetic recognizes that for many applications, such as machine learning, perfect precision is not required. The field has also seen the emergence of

quantum homomorphic encryption, a corresponding technology for delegated computation in quantum cloud networks.[31]

These research efforts on compilers, hardware, and new cryptographic schemes are not

isolated developments. They represent a collective and coordinated effort to address the performance and usability challenges of HE from multiple angles. This comprehensive approach underscores a profound transition in the field, moving from a purely academic pursuit to a concerted, engineering-focused endeavor aimed at making HE a practical reality.

7. Conclusion

Homomorphic Encryption stands as a groundbreaking cryptographic technology that addresses a fundamental tension in the digital age: the need to preserve data privacy while also harnessing data for valuable insights. By enabling computation directly on encrypted data, HE creates a new paradigm of "data in use" security, which is a prerequisite for unlocking new applications in sensitive domains like cloud computing, healthcare, and finance.

The field has evolved from foundational, but slow, theoretical schemes to specialized, application-optimized systems. This progression is a direct response to a well-defined technical problem: noise accumulation. The invention of bootstrapping for Fully Homomorphic Encryption solved this problem, transforming a theoretical concept into a functional technology. However, the computational overhead remains a significant barrier to widespread adoption.

The analysis indicates that the high latency and resource demands of current HE systems are not insurmountable limitations but are, in fact, the driving force behind a new wave of innovation. This is evidenced by the growing body of research dedicated to optimizing HE through the development of specialized compilers, hardware accelerators, and new, more efficient cryptographic schemes. This movement from pure theory to applied engineering signifies that homomorphic encryption is maturing and poised to become a foundational pillar of future data security, enabling a new era of secure, privacy-preserving computation.

Works cited

1. Technical Principles and Applications of Fully Homomorphic ..., accessed on September 5, 2025, https://www.gate.com/learn/articles/technical-principles-and-applications-of-fully-homomorphic-encryption-fhe/4517
2. How Homomorphic Encryption Works – Explained in Plain English - freeCodeCamp, accessed on September 5, 2025, https://www.freecodecamp.org/news/homomorphic-encryption-in-plain-english/
3. Homomorphic Encryption | CyberArk, accessed on September 5, 2025, https://www.cyberark.com/what-is/homomorphic-encryption/
4. Homomorphic encryption - Wikipedia, accessed on September 5, 2025,

https://en.wikipedia.org/wiki/Homomorphic_encryption
5. Privacy Tech-Know blog: Computing while blindfolded – Lifting the ..., accessed on September 5, 2025, https://www.priv.gc.ca/en/blog/20231024/
6. Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing, accessed on September 5, 2025, https://www.mdpi.com/2624-800X/3/1/4
7. Homomorphic encryption: a mathematical survey - EMS Press, accessed on September 5, 2025, https://ems.press/content/book-chapter-files/33149
8. What Is Homomorphic Encryption? Definition - Entrust, accessed on September 5, 2025, https://www.entrust.com/resources/learn/homomorphic-encryption-explained
9. Types of Homomorphic Encryption - IEEE Digital Privacy, accessed on September 5, 2025, https://digitalprivacy.ieee.org/publications/topics/types-of-homomorphic-encryption/
10. What Is Homomorphic Encryption? - Supermicro, accessed on September 5, 2025, https://www.supermicro.com/en/glossary/homomorphic-encryption
11. Differences FHE and PHE- Wodan AI - Secure AI, accessed on September 5, 2025, https://wodan.ai/differences-fhe-and-phe/
12. Homomorphic encryption for privacy-preserving computation - World Journal of Advanced Research and Reviews, accessed on September 5, 2025, https://wjarr.com/sites/default/files/WJARR-2020-0053.pdf
13. Advances and Applications in Fully Homomorphic Encryption Research | Applied and Computational Engineering, accessed on September 5, 2025, https://www.ewadirect.com/proceedings/ace/article/view/20959
14. Optimizing parameters for efficient computation with fully homomorphic encryption schemes - TÜBİTAK Academic Journals, accessed on September 5, 2025, https://journals.tubitak.gov.tr/cgi/viewcontent.cgi?article=4117&context=elektrik
15. Accelerating Homomorphic Encryption in RISC-V ... - AFIT Scholar, accessed on September 5, 2025, https://scholar.afit.edu/cgi/viewcontent.cgi?article=8784&context=etd
16. Homomorphic Technologies Could Process Still-Encrypted Data - Communications of the ACM, accessed on September 5, 2025, https://cacm.acm.org/news/homomorphic-technologies-could-process-still-encrypted-data/
17. Introduction to the BGV FHE Scheme - KFUPM, accessed on September 5, 2025, https://faculty.kfupm.edu.sa/coe/mfelemban/SEC595/References/Introduction%20to%20the%20BGV%20FHE%20Scheme.pdf
18. Homomorphic Encryption and Lattices - ASecuritySite.com, accessed on September 5, 2025, https://asecuritysite.com/lattice/bfv_bgv
19. CKKS.org, accessed on September 5, 2025, https://ckks.org/
20. CKKS explained: Part 1, Vanilla Encoding and Decoding ..., accessed on September 5, 2025, https://openmined.org/blog/ckks-explained-part-1-simple-encoding-and-decodi

ng/
21. Faster homomorphic comparison operations for BGV and BFV - ResearchGate, accessed on September 5, 2025, https://www.researchgate.net/publication/351159408_Faster_homomorphic_comparison_operations_for_BGV_and_BFV
22. Homomorphic Encryption Use Cases - IEEE Digital Privacy, accessed on September 5, 2025, https://digitalprivacy.ieee.org/publications/topics/homomorphic-encryption-use-cases/
23. What is homomorphic encryption for healthcare data? - Thought Leadership, accessed on September 5, 2025, https://www.healthcare.digital/single-post/what-is-homomorphic-encryption-for-healthcare-data
24. Homomorphic Encryption Is Changing the Healthcare Industry (Here's How) - InnoBoost SA, accessed on September 5, 2025, https://www.inno-boost.com/blog/homomorphic-encryption-in-healthcare/
25. Use Case: Preventing Financial Fraud Across Different Jurisdictions ..., accessed on September 5, 2025, https://fpf.org/resource/use-case-preventing-financial-fraud-across-different-jurisdictions-with-fully-homomorphic-encryption/
26. Is Homomorphic Encryption ready to solve the AI Privacy Problem? : r/cryptography - Reddit, accessed on September 5, 2025, https://www.reddit.com/r/cryptography/comments/1gf19wy/is_homomorphic_encryption_ready_to_solve_the_ai/
27. (PDF) Automatic Code Optimization in Fully Homomorphic ..., accessed on September 5, 2025, https://www.researchgate.net/publication/393405842_Automatic_Code_Optimization_in_Fully_Homomorphic_Encryption_Compilers
28. arxiv.org, accessed on September 5, 2025, https://arxiv.org/html/2312.14250v1#:~:text=Fully%20homomorphic%20encryption%20(FHE)%20can,as%20parameterization%20and%20circuit%20optimizations.
29. Google's Fully Homomorphic Encryption Compiler — A Primer - Math ∩ Programming, accessed on September 5, 2025, https://www.jeremykun.com/2023/02/13/googles-fully-homomorphic-encryption-compiler-a-primer/
30. (PDF) Advances and Applications in Fully Homomorphic Encryption ..., accessed on September 5, 2025, https://www.researchgate.net/publication/389418300_Advances_and_Applications_in_Fully_Homomorphic_Encryption_Research
31. Multi-party dynamic quantum homomorphic encryption scheme based on rotation operators, accessed on September 5, 2025, https://arxiv.org/html/2505.06955v1