The RISE of Fully Homomorphic Encryption

OFTEN CALLED THE HOLY GRAIL OF CRYPTOGRAPHY, COMMERCIAL FHE IS NEAR

MACHE CREEGER, CORNAMI INC.

FHE (fully homomorphic encryption) provides quantum-secure computing on encrypted data, guaranteeing that plaintext data and its derivative computational results are never exposed and remain secure from modification and/or breach despite compromised infrastructures. Most FHE schemes are based on lattice mathematics, are believed to be secure from breach by quantum computing,[9] and are considered PQC (post-quantum cryptography). New hardware accelerator architectures are an active

area of research and development,[1,3] and academic research continues to develop new and more efficient implementation schemes, making the full implication of FHE on data processing close to realization. With the advent of commercial FHE:

- Data, including its unrestricted computational derivatives, remains encrypted both at rest and throughout its life cycle and is decrypted to plaintext only in secure, trusted environments.
- Valuable insights through AI (artificial intelligence), big data, and analytics can be extracted from data—even from multiple and different sources—all without exposing the data, secret decryption keys, or, if need be, the underlying evaluation code.

CURRENT DATA SECURITY MODELS: NOT ONLY BROKEN BUT QUICKLY LOSING RELEVANCE

Industry-standard, perimeter-based security techniques common in today's IT infrastructure are built with thousands of integrated, constantly changing hardware and software components. They mostly depend on encryption techniques that rely on the difficulty of existing hardware to find discrete logarithms and/or factor large integers. This leads to some uncomfortable conclusions about the state of data security today:

- It is a statistical certainty that infrastructure breach points will always exist regardless of effort because of the number and ever-changing nature of these components. The only unknown is whether they have been identified and exploited. Protecting data has become an increasingly complex and breach-prone process because provable data

security is unachievable with this approach. Moreover, data processing is operating in an increasingly aggressive regulatory context with significant consequences and costs for breaches.[7]

- Widely used encryption technologies depend on the difficulty of finding discrete logarithms and/or factoring large integers on standard hardware. Quantum computing algorithms can easily solve these problems.[11] With the commercial quantum computing market growing at a compound annual growth rate of 36.5% and projected to reach $1,987.6 million by 2028[8] , these encryption techniques are quickly becoming obsolete and new PQC is needed.

What is required is a security mechanism that:

- Assumes that IT infrastructure is already compromised, so does not depend on a strong perimeter defense to protect data.
- Uses PQC encryption techniques not considered vulnerable to quantum computing attacks.

FHE meets both these requirements.

HISTORY OF HOMOMORPHIC ENCRYPTION

The idea of direct computation on encrypted data was first recognized in 1978 by Ronald L. Rivest, Len Adelman, and Michael L. Dertouzos.[10] They observed that under RSA (Rivest-Shamir-Adleman) encryption, two encrypted numbers could be multiplied and the result would be equivalent to the plaintext product encrypted using the same key. They called these properties *privacy homomorphisms*, recognizing that encryption schemes

can have properties where the results from a set of operations on plaintext data equals the results of those same operations performed on their encrypted forms and then decrypted. RSA encryption exhibited *multiplicative homomorphism*.

They recognized that:

- With HE (homomorphic encryption), the ability to do computation on encrypted data, data access can be separated from data processing by allowing computation to occur on encrypted data without the need for the secret decryption key.
- A user could take a piece of data, encrypt it homomorphically, use that encrypted data in a query (where the query itself could be encrypted or not) to a database, and get a result encrypted in the same way.
- At no time during this computation were the original query data, secret decryption key, results of the query, or—if encrypted—the query itself, ever exposed.

In 2009, more than 30 years later, Craig Gentry proposed the first plausibly secure FHE scheme.[4] Algorithms were defined as a circuit of logic gates, and unrestricted computation occurred on encrypted data with results encrypted in the same way. It was extremely slow, taking about 30 minutes to complete a single logic gate on standard x86 hardware.[5] Continued research has resulted in four distinct generations of FHE and substantial speed-up on standard hardware platforms.

Today, conventional wisdom suggests that an additional performance acceleration of at least another 1 million times would be required to make FHE operate at commercially viable speeds. At the moment, Cornami is the

only commercial chip company to announce a forthcoming product that can meet and exceed that performance level.[3]

FHE BUILDS ON PUBLIC-KEY ENCRYPTION

FHE provides all the functions supported by asymmetric PKE (public key encryption). As it is used today, PKE is based on finding discrete logarithms or factoring large integers and has five properties:

- *Key generation:* (sk, pk) ← K(λ) where key generation function K with argument random seed number λ produces a key pair consisting of a secret key sk and a public key pk.
- *Encryption:* c ← E(pk, m) where encryption function E with arguments pk and plaintext message m produces encrypted message ciphertext c.
- *Decryption:* m ← D(sk, c) where decryption function D with arguments sk and c produce m.
- *Correctness:* m = D(sk, E(pk, m)) for all key pairs, messages, and encryption randomness.
- *Semantic security:* ∀m ∈ {0, 1} – for all single-bit messages m, member of the set 0 and 1, E(pk, 0) and E(pk, 1) must be computationally indistinguishable and must be probabilistic (e.g., there should be many encrypted messages c per plaintext message m).

For use in HE, two more properties must be added:

- *Evaluate:* Along with the K, E, and D functions, V for evaluate is added.
- *Correctness:* D(sk, V(pk, f, c_1, … c_n)) = f(m_1, …, m_n) where decryption function D with arguments sk and evaluation function V with arguments pk; function f where f ∈ $\mathcal{F}$ (a set of or family of efficiently computable functions that have the desired homomorphic properties);

and ciphertexts $c_1, \ldots, c_n$ are equal to function f applied to arguments $m_1, \ldots, m_n$. For multiplication this would be D(sk, HE-MULTIPLY(pk, MULTIPLY, E(pk, m_1), E(pk, m_2))) = MULTIPLY(m_1, m_2).

To achieve unrestricted homomorphic computation, or FHE, you must choose $\mathcal{F}$ to be a set of functions that is complete for all computation (e.g., Turing complete). The two functions required to achieve this goal are bit Addition (equivalent to Boolean XOR) and bit Multiplication (equivalent to Boolean AND), as the set {XOR, AND} is Turing complete. Any computable function can be created with a combination of XOR and AND gates. If one homomorphically computes SUMs and PRODUCTs on encrypted bits, then one can compute *any* function on encrypted bits.

While bit XOR/Addition and bit AND/Multiplication are necessary for a homomorphic computational system to be Turing complete, algorithms need not be directly defined in those low-level terms. Current FHE models define computation in Boolean circuits, integer arithmetic, or real/complex arithmetic.

HE SECURITY

In their 1978 paper, Rivest, Adelman, and Dertouzos proposed that the secret key sk (they used the variable p) be hidden in a public key pk by creating random multiples of p (e.g. $q_i p$) where q_i is a secret factorization that is different for each encryption. Encryption of a single-bit b using a public key was the addition of random multiples of p to b. Decryption was then m = c modulo p modulo 2. Sadly, this approach breaks *semantic security* since

encryptions of plaintext bit 0 are just multiples of p:

1. $c = q_i p + b \text{ modulo } 2$
2. $= q_i p + 0 \text{ modulo } 2$
3. $p = GCD(\langle \text{encryptions of } 0 \rangle)$

In 2010 Martin van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan[12] (DGHV) determined that adding noise to the pq_i public key blocks GCD (greatest common divisor) secret-key discovery and, at present, any other secret-key discovery approach. The amount of noise to be added is determined by the *Approximate GCD assumption*: If you sample many integers from set $\{x_i = q_i p + 2r_i : r_i << p : p << q_i\}$ where (1) r_i is a slight amount of noise and is different for each encryption, and (2) each x_i is very close to multiples of p but not exact multiples of p, then the set of integers x_i is indistinguishable from random integers of the same size.

HE ENCRYPTION/DECRYPTION

To encrypt a bit *b*

HE encrypts a plaintext bit into a polynomial.

1. Pick a large, odd number p to be the secret key.
2. For each encryption, pick a random, large multiple of p, say $q_i p$.
3. Then for each encryption, sum bit b and $q_i p$ with a $Noise$ expression defined by the doubling of a random small number r_i to $2r_i$. This produces ciphertext $c = q_i p + 2r_i + b$ where $q_i p + 2r_i$ is the public key.

To decrypt ciphertext *c*

$b = c \text{ modulo } p \text{ modulo } 2$ removes the $Noise$.

$= q_i p + 2r_i + b \text{ modulo } p \text{ modulo } 2$

HE addition – XORing two encrypted bits

$$c_1 = q_1p + 2r_1 + b_1$$
$$c_2 = q_2p + 2r_2 + b_2$$
$$c_1 + c_2 = p(q_1 + q_2) + \underbrace{2(r_1 + r_2)}_{\textit{Noise}} + (b_1 + b_2)$$

HE multiplication – ANDing two encrypted bits

$$c_1 = q_1p + 2r_1 + b_1$$
$$c_2 = q_2p + 2r_2 + b_2$$
$$c_1c_2 = p(q_1q_2 + q_1b_2 + q_2b_1) + \underbrace{r_1(2pq_2 + b_2) + r_2(2pq_1 + b_1) + r_1r_2 + b_1b_2}_{\textit{Noise}}$$

Noise growth

- Addition: $2(r_1 + r_2)$
  $\text{Noise} = 2 \times \langle\text{initial noise}\rangle$
- Multiplication: $r_1(2pq_2 + b_2) + r_2(2pq_1 + b_1) + r_1r_2$
  $\text{Noise} = \langle\text{initial noise}\rangle^2$

Problem

If $|\text{Noise}|$ exceeds $p/2$, decryption cannot be guaranteed. Because Addition Noise growth is linear and Multiplication is exponential, if there is no mechanism to reset Noise growth, this approach will allow many additions and some multiplications before reaching the p/2 limit. Working within the p/2 Noise limit is the definition of SHE ("somewhat" homomorphic encryption) and can be effective for many valuable, bounded, use cases such as

database query and spam filtering. SHE does not have a mechanism to reset Noise growth during encrypted value computation, does not support unrestricted computation on encrypted data, and therefore is not FHE.

Resetting noise during HE computation to achieve FHE

Before Gentry's 2009 paper, Noise aggregated during HE computation significantly limited HE use cases. There were basically two options to address larger homomorphic computational threads.

Option 1

Increase the Noise limit by increasing the size of the secret key sk. This would increase the amount of computation that could be accomplished before hitting the sk/2 limit.

Option 2

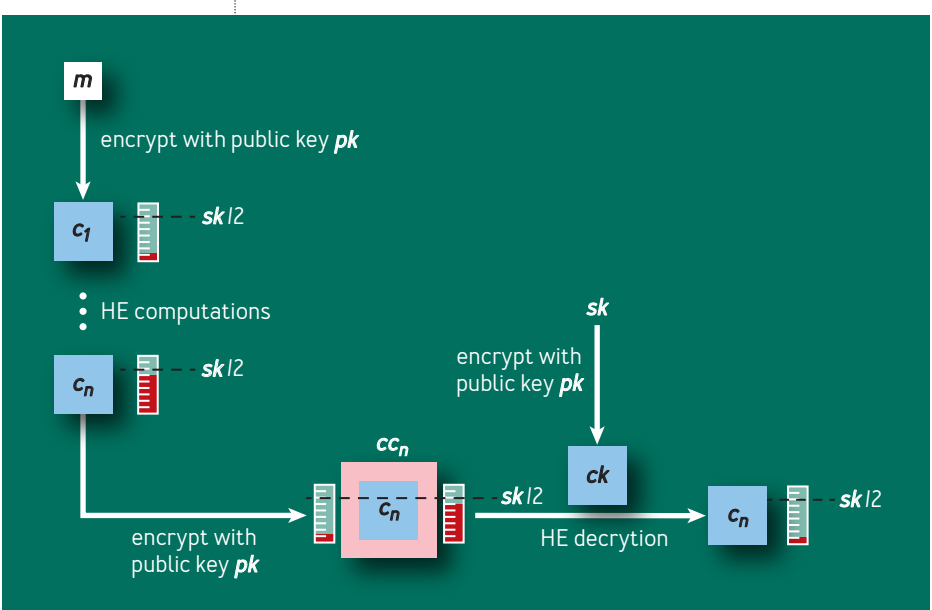
1. Freeze a homomorphic computational thread on an untrusted, insecure site.
2. Transfer the encrypted intermediate value c_n back to a secure, trusted site.
3. Decrypt c_n with the secret key sk to plaintext m_n.
4. Encrypt m_n back to c_n with a public key pk reducing Noise to a nominal state.
5. Transfer c_n back to the untrusted, insecure site.
6. Restart the homomorphic computational thread with the new re-encrypted and lower noise c_n.

Obviously, Option 2 is not practical. Gentry developed a mechanism to reset Noise in an encrypted result so that a computational thread could continue indefinitely. In his approach he used lattice-based cryptography that

is considered resistant to attack by both classical and quantum computer methods.

Gentry effectively implemented Option 2 by applying a recursive, embedded, homomorphic decryption that allowed for the noise of encrypted value c_n to be reset without exposing it or the secret key to potential breach or physically transferring it to a secure, trusted site for decryption and re-encryption. In this way Gentry showed that unrestricted computation on encrypted data or FHE is possible. Gentry's approach follows these steps (as shown in figure 1):

FIGURE 1: **UNRESTRICTED COMPUTATION ON ENCRYPTED DATA**



1. Encrypt a plaintext message m with public key pk to produce ciphertext c_1.
2. Perform some number of HE computations on c_1 to produce c_n such that c_n is near to but not over the noise limit $sk/2$.
3. Encrypt secret key sk with public key pk to create an encrypted secret key ck.
4. Encrypt c_n with public key pk, resulting in a new double-encrypted cc_n.
5. HE Decrypt cc_n using encrypted secret key ck to produce c_n with a reset noise level.
6. Continue the computation using c_n.

What Gentry achieved was to decrypt and re-encrypt an encrypted value c using a homomorphic computation with an encrypted secret key sk and the public key pk. Gentry called his $Noise$ reset process *bootstrapping*. While it showed that unrestricted computation on encrypted data is possible, two significant limitations prevented its use in programming applications: (1) The computation required by the bootstrapping algorithm far exceeded the performance capabilities of available hardware platforms; (2) it lacked an efficient implementation of *conditionals,* which enable programmatic comparison and selection/jump operations.

Since 2009, researchers have evolved substantial performance and functional improvements over the original Gentry scheme: Enhance overall homomorphic computational performance; increase bootstrap performance; decrease the number of bootstraps required for a fixed amount of homomorphic computation; minimize

noise growth during homomorphic computation without bootstrapping; and refine cryptographic models based on known, high-difficulty problems in lattice mathematics not solvable with quantum computing.

The following are some of the more notable advances:

- *LWE (learning with errors) and RLWE (ring learning with errors).* Equivalent to solving the CVP (closest vector problem) in lattice mathematics and based on the inability of determining coefficients (which represent the secret key) in a sampling of a system of linear equations (LWE) or polynomial rings over finite fields (RLWE), where each equation has a small, random, additive error.
- *Leveling.* Allowing the evaluation of a logic-gate circuit of predetermined depth before requiring a bootstrap.
- *Relinearization.* Computationally expensive method of reducing both homomorphic computational cost and storage burden by reducing cyphertext length (resulting from homomorphic multiplication) while preserving correctness of the underlying message.
- *Modulus switching.* Reducing noise without secret-key use while preserving the integrity of a ciphertext c by dividing a ciphertext $c \text{ modulo } q$ by noise factor $|r|$ to produce a new, lower-noise, equivalent cyphertext $c' = \frac{c}{r} \text{ modulo } \frac{q}{r}$.

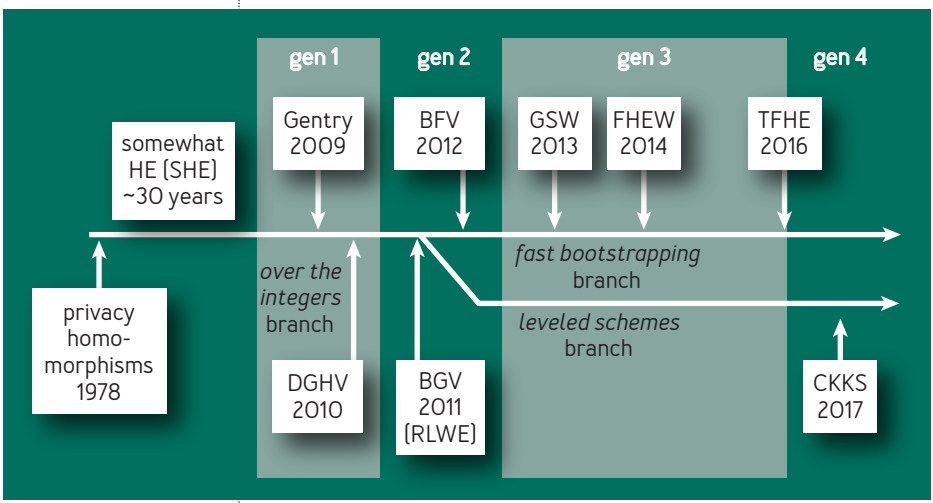
FHE GENERATIONS

The history of FHE is shown in figure 2.[13]

First generation

Gentry's original ideal lattice-based FHE scheme supported both addition and multiplication on cyphertexts

FIGURE 2: **FHE GENERATIONS**



that allowed for logic circuits to perform unrestricted computation (e.g., Turing complete). It was very slow. DGHV then replaced the SHE portion of Gentry's approach with a simple integer-based scheme.

Second generation

BFV (Brakerski/Fan-Vercauteren) and BGV (Brakerski-Gentry-Vaikuntantan) introduced LWE and RLWE security models and also introduced leveling schemes that allow for the execution of a logic-gate circuit of set depth before requiring a bootstrap.

Third generation

GSW (Gentry-Sahai-Waters) avoided computationally expensive *relinearization* used in homomorphic

multiplication. It had slower noise growth. There was development of more efficient ring variants with FHEW (Ducas-Micciancio – "Fastest Homomorphic Encryption in the West"), as well as simplification and increased optimization of bootstrapping.

Fourth generation

CKKS (Cheon-Kim-Kim-Song) introduced efficient rounding operations for encrypted values that control noise rate increases in HE multiplication and reduce the number of bootstraps in a logic circuit. It also introduced the concept of PBS (programmable bootstrapping) to TFHE[2] (torus fully homomorphic encryption), reducing the number of bootstraps required in a logic circuit.

FHE MODELS OF COMPUTATION

Current FHE schemes[6] implement computation in one of three ways:

Boolean circuits

- Plaintext: bits
- Computation: arbitrary Boolean logic-gate circuits
- Fast number comparison and bootstrapping
- GSW, FHEW, TFHE

Exact/modular arithmetic

- Plaintext: integers modulo a plaintext modulus a (or their vectors)
- Computation: integer arithmetic circuits mod a
- Efficient SIMD (single-instruction multiple data) batch computations over integer vectors

- Fast, high-precision integer arithmetic and scalar multiplication
- Leveling that can avoid bootstrapping
- BGV, BFV

Approximate number arithmetic

- Plaintext: real or complex numbers
- Computation: similar to floating-point arithmetic
- Fast polynomial approximation
- Relatively fast multiplicative inverse and discrete Fourier transform
- Deep approximate computations, such as logistic regression learning
- Efficient SIMD batch computations over real-number vectors
- Leveling that can avoid bootstrapping
- CKKS

Table 1 shows current FHE frameworks and supported schemes, and table 2 lists FHE platforms.

FHE USE CASES

With hardware FHE accelerators now on the horizon, it is important to understand some of the use cases that will become available with commercial-grade FHE. This is a partial list of potential application areas:

- Secure data from breach/modification throughout its life cycle. Privacy-preserving computing on encrypted data guarantees that data and its derivative computational results remain secure from modification and/or breach despite compromised infrastructure. Provable security

TABLE 1: **CURRENT FHE FRAMEWORKS AND SUPPORTED SCHEMES**

FRAMEWORK	DEVELOPER	BGV	CKKS	BFV	FHEW	CKKS BOOT-STRAPPING	TFHE
HElib	IBM	✖	✖				
Microsoft SEAL	Microsoft		✖	✖			
PALISADE	Duality and a DARPA consortium	✖	✖	✖	✖		✖
HEAAN	Seoul National University		✖			✖	
FHEW	Leo Ducas and Daniele Micciancio				✖		
TFHE	Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachene, and Zama (Concrete)						✖
FV-NFLlib	CryptoExperts			✖			
NuFHE	NuCypher						✖
Lattigo	EPFL-LDS		✖	✖		✖	

for data both at rest and throughout its computational life cycle will accelerate the move to untrusted platforms for computation services on confidential data, removing much of the rationale for private data-center use.

- Secure data from quantum computing attacks. The lattice-based PQC used in FHE is not considered vulnerable to quantum computing attacks, while industry-standard number factoring and discrete logarithms-based encryption used today are vulnerable. With many quantum computer products either released

TABLE 2: **FHE PLATFORMS**

INPHER	MPC supported XOR platform. Will include FHE when hardware acceleration becomes available
ZAMA	Concrete is a continual evolution of TFHE by adding Programmable Bootstrapping (PBS) feature
DUALITY	SecurePlus products uses HE to support Analytics and Machine Learning, SQL-like Query, and Collaboration. Palisade supported platform.
ENVEIL	ZeroReveal platform based on HE. Supports Database Query and Machine Learning
IBM	IBM Security Homomorphic Encryption Services and HElib Library
GOOGLE	FHE C++ Transpiler
MICROSOFT	Microsoft SEAL is an open-source homomorphic encryption library
CORNAMI	Developing a semiconductor chip-based scalable fabric of processor cores that can execute FHE at any speed including real-time.

or scheduled to be released, the PQC era has already started.

- Protect application services data, results, and analytic models from disclosure. Execution of big data, AI, and/or analytic services using FHE-secure, user-encrypted data inputs, service results, and analytic model information (such as neural network weights).
- Analysis of confidential data aggregated from multiple organizations. Different confidential datasets from multiple organizations are aggregated and analyzed without underlying data disclosure. Examples include (1) big data, AI, or analytics insights into industrywide trends; (2) balance-sheet aggregation to evaluate mergers and acquisitions; (3) combining data from

different vendors to facilitate pharmaceutical drug trials; and (4) analysis to be applied to the combined potential partnership data to determine potential business value.

- Secure and confidential rule matching against network traffic. Patterns of behavior, methods, and techniques used by network bad actors are learned over time through advanced NTA (network traffic analysis), are defined as rules, and encrypted using FHE. Those encrypted rules are applied using FHE computation to network traffic in untrusted environments identifying and monitoring threat actors' presence without revealing threat signatures or matching traffic. This is useful in both WAN/LAN computer network security and AML (anti-money laundering).
- PSI (private set intersection): Secure and confidential data intersection check within a larger database. This is the ability to query if specific data exists in a larger data store without revealing information about the contents of the query or the data store.
- FHE-enhanced blockchains. Enabling *private transactions* recorded on a blockchain using FHE and ZKP (zero-knowledge proofs) can prove that a transaction occurred without revealing data detail.
- Guaranteed data security and integrity across sensor/controller/actuator realtime control chain. Through encryption of sensor data at the source and supporting encrypted computation throughout the realtime control chain, data can be protected from both breach and modification.
- Monetization of confidential data resources. Revenue

streams can be produced from licensing FHE-encrypted proprietary/confidential datasets for use by untrusted machine learning/big data/analytic applications on untrusted platforms.

SUMMARY

Often called the Holy Grail of cryptography, commercial FHE is near. Strong forces are combining to drive FHE to commercial reality. These include the following:

- Provable security models for IT infrastructures will become an unavoidable requirement.
- Increased regulatory requirements and the widespread availability of quantum computers will make PQC an imperative for government and industry.
- Aggregation of confidential data from multiple sources without its disclosure will open up far-reaching ways for organizations to partner and extract new high-valued insights into a wide range of industry trends.
- High-value analytic service models will be protected from disclosure even when operating on untrusted platforms.
- Data integrity will be maintained through the sensor-processing-actuator chain.
- Opportunities will exist for new data-licensing revenue models that do not risk confidential data disclosure.
- Active research and development will continue to develop hardware FHE accelerators.
- Continued academic research will accelerate FHE scheme performance.

Once commercial FHE is achieved, data access will become completely separated from unrestricted data

processing, and provably secure storage and computation on untrusted platforms will become both relatively inexpensive and widely accessible. In ways similar to the impact of the database, cloud computing, PKE, and AI, FHE will invoke a sea change in how confidential information is protected, processed, and shared, and will fundamentally change the course of computing at a foundational level.

References

1. Atherton, K. 2021. DARPA awards contracts for encrypted data processing. Breaking Defense; https://breakingdefense.com/2021/03/darpa-awards-contracts-for-encrypted-data-processing/.
2. Chillotti, I., Joye, M., Paillier, P. 2020. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks; https://whitepaper.zama.ai/.
3. Cornami, Inc. Fully homomorphic encryption; https://cornami.com/fully-homomorphic-encryption-fhe/.
4. Gentry, C. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 169–178; https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf.
5. Gentry, C., Halevi, S. 2011. Implementing Gentry's Fully-Homomorphic Encryption Scheme; IBM Research. https://eprint.iacr.org/2010/520.pdf.
6. Homomorphic Encryption Standardization Consortium. 2018. Building Applications with Homomorphic Encryption; https://homomorphicencryption.org/wp-content/uploads/2018/10/CCS-HE-Tutorial-Slides.pdf.
7. IT Governance. 2018. Data breach notification laws by

state; https://www.itgovernanceusa.com/data-breach-notification-laws.
8. MarketWatch. Quantum Computing Market Size In 2022; https://www.marketwatch.com/press-release/quantum-computing-market-size-in-2022-352-cagr-analysis-of-key-trends-with-top-countries-data-top-key-manufactures-industry-dynamics-insights-and-future-growth-2028-exclusive-110-pages-report-2022-07-21.
9. Regev, O. 2006. Lattice-based cryptography. In *Advances in Cryptology*, ed. C. Dwork, 131–141. Lecture Notes in Computer Science 4117. Berlin, Heidelberg: Springer; https://link.springer.com/content/pdf/10.1007/11818175_8.pdf.
10. Rivest, R. L., Adleman, L., Dertouzos, M. L. 1978. On data banks and privacy homomorphism. Massachusetts Institute of Technology. Academic Press; https://people.csail.mit.edu/rivest/RivestAdlemanDertouzos-OnDataBanksAndPrivacyHomomorphisms.pdf.
11. Shor, P. W. 1994. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 124–134; https://klein.mit.edu/~shor/papers/algsfqc-dlf.pdf.
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V. 2010. Fully homomorphic encryption over the integers. In *Advances in Cryptology – Eurocrypt 2010*, ed. H. Gilbert. Lecture Notes in Computer Science 6110. Berlin, Heidelberg: Springer; https://eprint.iacr.org/2009/616.pdf.
13. Wikipedia. Homomorphic encryption; https://en.wikipedia.org/wiki/Homomorphic_encryption.

Mache Creeger *has spent his career focused on commercializing newly emerging technologies and has worked at Sun, MIPS, Sony, LISP Machine Inc., InstallShield, several startups, Naval Research Lab, and Jet Propulsion Lab, and ran his own consulting practice. At Cornami, he is VP of business development and focuses on working with FHE (fully homomorphic encryption) framework vendors to enable Cornami hardware acceleration of FHE applications. Creeger founded and ran* acmqueue*'s CTO Roundtable series as head wrangler, was a columnist for* acmqueue *magazine, and co-chaired three panel discussions about emerging commercial technology trends (synthetic biology, human augmentation, and autonomous vehicles) as a volunteer at the Stanford Chapter of the MIT Enterprise Forum (VLAB). He has written 23 published articles and has one US patent application. He has a B.S. in psychology and biochemistry, one year of postgraduate study in biochemistry, and an M.S. in computer science, all from the University of Maryland.*