Peter Gutmann, pgut001@cs.auckland.ac.nz | David Naccache, david.naccache@ens.fr | Charles C. Palmer, ccpalmer@us.ibm.com

Fully Homomorphic Encryption:

Computations with a Blindfold

Marc Beunardeau, Aisling Connolly, Rémi Géraud, and David Naccache | École normale supérieure

Today's focus on mobility has changed the way we use technology and shifted our concerns. To access data from everywhere, businesses and individuals alike are outsourcing their data, and sometimes computations, to the cloud—external servers managed by other companies. This move comes with a serious concern: the cloud service providers themselves can't be trusted. Market competition is ripe, making "extras" like security and performance expensive. More important, the information stored on cloud services is increasingly valuable and personal. If hackers took control of the servers, they could swim freely in an ocean of extremely sensitive data.

Over the past 50 years, the blossoming science of cryptography has provided clever and powerful methods to secure private information. In recent years, a technological shift and a revolutionary idea have combined to bring us to the threshold of outsourced, secure computation: *fully homomorphic encryption* (FHE).

Motivation: Uprooting to the Cloud

Careful users might encrypt their data before uploading it to an insecure cloud service. This works if users intend to only use cloud services for storage. But they increasingly want cloud services to provide actual services, including computing on data. For instance, they might want to send a query to a search engine, which would run algorithms on the data and return a result. However, encrypting data makes it impossible for cloud services to use it (see Figure 1).

To leverage a service providers' computing power (which you often want to do if you use a smartphone), you can't encrypt your data in the traditional way. But if you don't encrypt your data, anyone (including the cloud service itself) can easily read it. Ronald Rivest and his colleagues first posed this dilemma in 1978,[1] and it remained unresolved for more than 30 years. In 2009, Craig Gentry finally provided the first satisfying answer.[2] It turns out that conciliating privacy and cloud computing is possible: enter FHE, which lets you outsource computation in a provably secure way.

Homomorphic Encryption

Let's start with a simple example of homomorphic encryption: concatenation. Concatenation entails putting two messages or two ciphertexts in sequence. If you're using a substitution cipher, concatenation is possible on ciphertexts; that is, you can concatenate and then encrypt, or encrypt and then concatenate—the result will be the same. Thus, you can securely outsource concatenation.

Truth be told, concatenation isn't an interesting computation. But a cryptosystem that allows arbitrary computations on encrypted

1540-7993/16/$33.00 © 2016 IEEE Copublished by the IEEE Computer and Reliability Societies January/February 2016

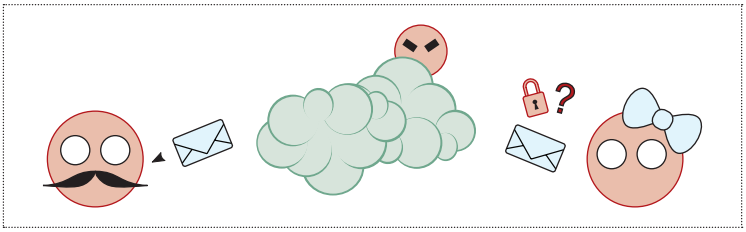

Figure 1. If Alice uses a cloud service to send a message to Bob, an eavesdropper (or the cloud service itself) could access her data. Alice could use encryption—but then she wouldn't be able to leverage the cloud service's computing power. So, at first glance, it seems like she might have to choose between maintaining her privacy and using the cloud.

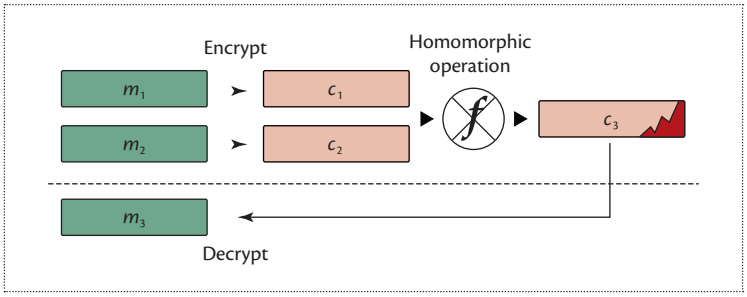

Figure 2. An illustration of homomorphic encryption. Alice encrypts messages m_1 and m_2. The cloud service then performs an operation on ciphertexts c_1 and c_2, yielding new ciphertext c_3 (with some noise, shown in red). Finally, Alice can decrypt c_3 to yield $m_3 = f(m_1, m_2)$.

data—called *homomorphic encryption*—would solve the dilemma. (We might be interested in other properties as well, such as whether the ciphertext remains the same size after operations, so that it resembles other ciphertexts.) To achieve homomorphic encryption, you need to perform two operations: addition and multiplication.

Somewhat Homomorphic Encryption

Many partially homomorphic schemes exist. Even one of the first public-key cryptosystems, RSA, lets you compute the sum of two plaintexts without decrypting.[3] (This is generally considered a vulnerability and is thwarted by so-called RSA padding functions.) Taher ElGamal's cryptosystem lets you compute a message product.[4] For a long time, we had to choose between addition and multiplication, and neither operation was sufficient for arbitrary computations.

The first major advance was the discovery of *somewhat homomorphic* encryption schemes.[5] Such schemes allow for arbitrary computations; however, they are "somewhat" homomorphic in the sense that each time you perform an operation on a ciphertext, the operation inevitably adds some noise (see Figure 2). A little noise doesn't affect decryption. But you can only perform so many operations before the accumulated noise renders the ciphertext undecipherable.

In other words, every time you perform an operation, there's a cost, and you can't exceed the fixed initial total cost. That being said, somewhat homomorphic encryption is still useful. For simple operations, noise simply vanishes on decryption.

For almost 20 years, the search continued for an encryption that wasn't plagued by noise. Many ideas were proposed, but all turned out to be insecure. But then Gentry's revolutionary insight finally opened the door to FHE. Table 1 illustrates the progression from partially to fully homomorphic encryption.

Fully Homomorphic Encryption

Gentry's idea was to look for an encryption scheme with *low decryption complexity*—decryption that requires very few operations.[2] Why does this matter? Because the decryption procedure itself can be computed (somewhat) homomorphically (see Figure 3). Such encryption schemes are called *bootstrappable*.

The intuition is that decryption removes the noise. Indeed, if we completely decrypt a ciphertext, we can "refresh" it simply by generating a new ciphertext that encrypts the same plaintext. However, we want a way to refresh that doesn't require a secret key. Bootstrapping decrypts the ciphertext homomorphically. "Blindly" decrypting and reencrypting a ciphertext provides a fresh, noise-free ciphertext on which we can perform additional computations ad libitum. It's actually a theorem:[2]

> If E is bootstrappable, then, for any integer d, one can construct a scheme $E^{(d)}$ that can evaluate any circuit (consisting of NAND gates) of depth d. The decryption circuit for $E^{(d)}$ is the same as for E, and the complexity of encryption is also the same.

The first step in FHE is to construct a bootstrappable somewhat homomorphic scheme. Gentry chose lattice encryption, because it's simple to decrypt with a shallow circuit. He also used a trick called "squashing" to make ciphertexts easier to decrypt. Most homomorphic constructions today rely on lattices, although alternatives exist. The decryption procedure for

Table 1. Comparison of partially, somewhat, and fully homomorphic encryption.

Type of homomorphic encryption	Unlimited no. of operations?	All possible computations?
Partially	Yes	No
Somewhat	No	Yes
Fully	Yes	Yes

lattices includes only simple arithmetic operations.

The second step is to construct the refresh, or bootstrapping, procedure. In essence, it's decryption performed homomorphically with an encrypted secret key. By decrypting, we effectively remove the noise. By doing so homomorphically, we reintroduce some noise (but less than we removed). Note that revealing the encrypted secret key doesn't compromise security—bootstrapping is a public operation. This double encryption is the starting point from which we can progressively clean noise from ciphertexts.

Of course, our goal is not merely to obtain refreshed encryptions of the same plaintext. As Gentry showed, if we can evaluate a NAND augmentation of the decryption circuit (that is, two copies of the decryption circuit connected to a NAND gate), we can generate an encryption of plaintext messages 1 and 2 (m_1 NAND m_2) under public key 2 (pk_2) by using the encrypted secret key (sk_1 under pk_2) together with the two ciphertexts encrypting m_1 and m_2, respectively, under pk_2. By recursively performing this operation on all ciphertexts at a given level in the circuit, we can evaluate a d-depth circuit of NANDs.

It's critical to ensure that the noise doesn't grow too much during homomorphic decryption, because we need the decryption to work correctly and allow further operations. So although we might need to bootstrap quite often, if we choose the correct parameters, FHE will work.

We can securely perform arbitrary complex computations, leveraging an external server's computational power. To illustrate this, imagine blind cooks (or photographers, silversmiths, and so on) using their skills on something they can't see. In this analogy, the cooks are the powerful cloud services and the raw material they work on is your data.

An open question remains: Can a (fully) homomorphic encryption scheme be constructed without starting from a somewhat homomorphic encryption scheme?

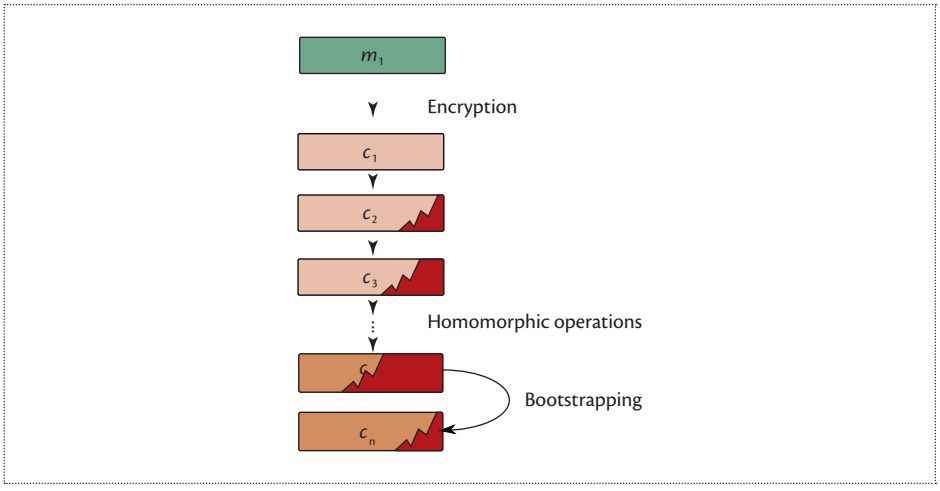

Figure 3. Noise builds up after each homomorphic operation. At some point, the noise will overrun the ciphertext and make the message undecryptable. Noise accumulation plagued somewhat homomorphic schemes by limiting the number of allowed operations. Craig Gentry used bootstrapping to solve this issue: by partially decrypting the ciphertext, he found that he could remove some of the noise and continue computing indefinitely.[2]

Limitations and Generations

So why isn't FHE already being used everywhere? Because although it works, FHE as described above isn't practical. At all.

Boostrapping operations are extremely costly in terms of computation and must be performed often. Moreover, ciphertexts are very large. In an early implementation by Gentry and Shai Halevi, the public key size ranged from 70 Mbytes for the "small" setting to 2.3 Gbytes for the "large" setting; the time to run one bootstrapping operation ranged from 30 seconds for the small setting to 30 minutes for the large setting.[6] Nevertheless, their approach showed FHE's feasibility and led to renewed efforts.

The security of the first generation of schemes following Gentry and Halevi's construction relied on the somewhat homomorphic component.[2,7,8] The efficiency of Gentry's and Halevi's implementation also garnered attention, leading to studies focused on reducing the scheme's complexity.[6,9,10] The following generation used different techniques to improve

efficiency.[11–13] In 2015, a variant on a scheme by Gentry and his colleagues[13] resulted in improvements in HElib (a software library that implements homomorphic encryption; https://github.com/shaih/HElib) that allowed bootstrapping in 0.61 seconds on a 3-GHz machine. In comparison, RSA encryption and decryption with a 1,024-bit key can be performed in less than a millisecond on the same machine.

Here we come across another open question: Can a noiseless (and, hence, very efficient) homomorphic scheme be constructed?

The Future of FHE

The path ahead will require more than just improved techniques. Faster FHE is useful. And FHE obviously has many applications, be it in hospitals, financial institutions, advertising, consulting, or pricing. FHE could even yield new uses for technology, such as secure lending of supercomputer time.

But FHE also opens the door to exciting new research. In 2013, Sanjam Garg and his colleagues used FHE to construct another mythical object: the first candidate multilinear map.[14,15] Essentially, multilinear maps are generalizations of the bilinear pairings that led to the pairing-based cryptography revolution of the early 2000s. More precisely, a (symmetric) κ-multilinear map is a nondegenerate map $e: \mathbb{G}^{\kappa} \rightarrow \mathbb{G}_T$ (where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order, denoted additively, and g is a generator of $\mathbb{G}$) such that, for all $a_1, \ldots, a_{\kappa} \in \mathbb{Z}_p$,

$$e(a_1 \cdot g, \ldots, a_{\kappa} \cdot g) = (a_1 \cdots a_{\kappa}) \cdot e(g, \ldots, g).$$

Although Garg and his colleagues' particular construction didn't work (it was insecure), it initiated a volley of new candidates. Research on FHE-powered multilinear maps has spurred new cryptographic applications; chief among them is the first proposed approach to general program obfuscation.[16] Cryptographic multilinear maps would drastically change cryptography's landscape.

Other FHE applications include functional encryption, verifiable computing, secure multiparty computation, and attribute-based encryption (ABE).[17] Let's look more closely at ABE, which lets you encrypt messages that can be read only by people with a set of attributes that are specified at encryption time. For example, imagine that you work for a company concerned about spies. The company requires each department to encrypt its monthly reports so that only the accounting department executives and the CEO can read them. Traditionally, this would mean that the company would have to set up a complicated key exchange protocol and deploy a secure key distribution infrastructure. If new accounting executives were appointed, transmitting their public keys to all employees would be unwieldy. But with ABE, the company could give new accounting executives the attribute, and each department would require only the "(accounting AND executive) OR CEO" attribute to be verified for decryption.

So the road toward secure cloud computing goes on. FHE has opened up a world of possibilities, and more applications will be discovered as implementations and schemes become more efficient. ■

References

1. R.L. Rivest, L. Adleman, and M.L. Dertouzos, "On Data Banks and Privacy Homomorphisms," R.A. DeMillo et al., eds., *Foundations of Secure Computation*, Academia Press, 1978, pp. 169–179; http://people.csail.mit.edu/rivest/pubs/RAD78.pdf.
2. C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proc. 41st Ann. ACM Symp. Theory of Computing* (STOC 09), 2009, pp. 169–178.
3. R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signature and Public-Key Cryptosystems," *Comm. Assoc. Computing Machinery*, vol. 21, no. 2, 1978, pp. 120–126.
4. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory*, vol. 31, no. 4, 1985, pp. 469–472.
5. D. Boneh, E.J. Goh, and K. Nissim, "Evaluating 2-DNF Formulas on Ciphertexts," *Proc. 2nd Theory of Cryptography Conf.* (TCC 05), LNCS 3378, 2005, pp. 325–341.
6. C. Gentry and S. Halevi, "Implementing Gentry's Fully-Homomorphic Encryption Scheme," *Proc. 30th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques* (EUROCRYPT 11), 2011, pp. 129–148.
7. M. van Dijk et al., "Fully Homomorphic Encryption over the Integers," *Proc. 29th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques* (EUROCRYPT 10), LNCS 6110, 2010, pp. 24–43.
8. Z. Brakerski and V. Vaikuntanathan, "Efficient Fully Homomorphic Encryption from (Standard) LWE," *Proc. 52nd Ann. Symp. Foundations of Computer Science* (FOCS 11), 2011, pp. 97–106.
9. N.P. Smart and F. Vercauteren, "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes," *Proc. 13th Int'l Conf. Theory and Practice of Public Key Cryptography* (PKC 10), LNCS 6056, 2010, pp. 420–443.
10. D. Stehlé and R. Steinfeld, "Faster Fully Homomorphic Encryption," *Proc. 16th Int'l Conf. Theory and Application of Cryptology and Information Security* (ASIACRYPT 10), LNCS 6477, 2010, pp. 377–394.
11. C. Gentry and S. Halevi, "Fully

Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits," *Proc. 52nd Ann. Symp. Foundations of Computer Science* (FOCS 11), 2011, pp. 107–109.
12. Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption without Bootstrapping," *Proc. 3rd Innovations in Theoretical Computer Science Conf.* (ITCS 12), 2012, pp. 309–325.
13. C. Gentry, A. Sahai, and B. Waters, "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based," *Proc. 33rd Ann. Int'l Cryptology Conf.* (CRYPTO 13), LNCS 8042, 2013, pp. 75–92.
14. S. Garg, C. Gentry, and S. Halevi, "Candidate Multilinear Maps from Ideal Lattices," *Proc. 32nd Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques* (EUROCRYPT 13), LNCS 7881, 2013, pp. 1–17.
15. D. Boneh and A. Silverberg, "Applications of Multilinear Forms to Cryptography," *Contemporary Mathematics*, vol. 324, 2003, pp. 71–90.
16. S. Garg et al., "Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits," *Proc. 54th Ann. Symp. Foundations of Computer Science* (FOCS 13), 2013, pp. 40–49.
17. A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *Proc. 24th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques* (EUROCRYPT 05), 2005, pp. 457–473.

Marc Beunardeau is a PhD candidate at the École normale supérieure. Contact him at marc.beunardeau@ingenico.com.

Aisling Connolly is a PhD candidate at the École normale supérieure. Contact her at aisling.connolly@ens.fr.

Rémi Géraud is a PhD candidate at the École normale supérieure. Contact him at remi.geraud@ens.fr.

David Naccache is a professor of computer science at the École normale supérieure. Contact him at david.naccache@ens.fr.