Fully Homomorphic Encryption: Cryptography's Holy Grail

David J. Wu

For over 30 years, cryptographers have embarked on a quest to construct an encryption scheme that would enable arbitrary computation on encrypted data. Conceptually simple, yet notoriously difficult to achieve, cryptography's holy grail opens the door to many new capabilities in our cloud-centric, data-driven world.

The advances in statistical and computational methods for machine learning coupled with the advent of powerful, cloud-based computing platforms in the last decade have ushered in the era of "big data." Not only are data more easily accessible than ever before, we now have available a wide range of tools that enable us to better understand and learn from these rich troves of data. However, as we transition into this cloud-based, data-centric environment, we often find ourselves revealing details of our personal lives to the cloud, be they in the form of movie preferences for recommender systems such as Netflix, or financial information for tax-preparation services such as TurboTax. Two natural question that arise are whether we can trust the cloud with our information and whether the notions of data-driven, cloud-based computing are inherently at odds with individual privacy.

Homomorphic Encryption

A simple and efficient solution for preserving the privacy of user information in cloud-based services is to encrypt the data that is sent to the cloud. However, this simple solution has a significant drawback in that if the data is encrypted using a conventional encryption algorithm (for example, using the AES block cipher), then the cloud is unable to operate on the data without needing to first decrypt. Of course, if we then share with the cloud the secret decryption key, we are back to square one where we have no privacy guarantees on the data. Thus, we ask whether there exist encryption schemes that allow some computation to be performed directly on encrypted data (*without* first decrypting it).

To answer this question, we begin in 1977 with the dawn of modern public-key cryptography. A public-key encryption scheme is specified by three algorithms: Setup, $\mathrm{Enc_{pk}}$, and $\mathrm{Dec_{sk}}$. As the names suggest, the Setup algorithm generates a public key pk and a secret key sk. The $\mathrm{Enc_{pk}}$ algorithm takes a message and outputs its encryption under the public key. The $\mathrm{Dec_{sk}}$ algorithm takes a ciphertext produced by the encryption function and decrypts it using the secret key to produce a message. The public-key nature of the encryption scheme means that anyone can encrypt a message using the public key, but only the holder of the secret key can decrypt.

For an encryption scheme to be useful, we require first a basic *correctness* property, that is, if one encrypts a message under the public key, decrypting the resulting ciphertext with the secret key should yield the original message. Expressed in our notation, we require that for all messages m, $\mathrm{Dec_{sk}}\left(\mathrm{Enc_{pk}}(m)\right) = m$. The more important requirement is the *security* requirement. We use the notion of semantic security against chosen plaintext attacks (CPA security) [1]. An encryption scheme is CPA-secure if no polynomial-time algorithm can distinguish an encryption of a message m_0 from an encryption of a message m_1 for all messages m_0 and m_1. Informally, the security requirement captures the notion that all ciphertexts look the same to an adversary, and thus, do not reveal any information about the underlying messages.

By themselves, the setup, encryption, and decryption algorithms do not enable computation on ciphertexts. To support computations on ciphertexts, we require additional structure in our ciphertext space. Consider an example: suppose we have an encryption scheme whose plaintext space is the additive group $\mathbb{Z}_n$ (the integers taken modulo n). Additionally, suppose we have encryptions c_0 and c_1 of two messages $m_0, m_1 \in \mathbb{Z}_n$, respectively. We say that this encryption scheme is *additively homomorphic* if there is an efficiently-computable operator $\star$ on ciphertexts such that $c_0 \star c_1$ is a valid ciphertext that decrypts to the sum $m_0 + m_1$. If such an operator $\star$ exists, then the encryption and decryption functions are group homomorphisms, hence the name *homomorphic* encryption. In words, in an additively homomorphic encryption scheme, given two ciphertexts, it is easy to construct a ciphertext that encrypts the sum of the underlying plaintext values. Though the existence of the $\star$ operator necessitates additional structure on the ciphertexts, an additively homomorphic encryption scheme must still satisfy the correctness and security requirements described earlier.

It might seem unlikely for a scheme to be both homomorphic and semantically secure; the former requirement necessitates sufficient structure on the ciphertexts to carry out some limited computation, while the latter requirement necessitates enough randomness in the ciphertexts so that an adversary is unable to distinguish between ciphertexts. However, it turns out that both of the earliest candidates for public-key cryptography, the ElGamal [2] and RSA [3] encryption schemes, are both homomorphic with respect to multiplication. As a concrete example, we describe the ElGamal encryption scheme here and check the homomorphic property. Let $\mathbb{G}$ be a group with prime order p and generator g. In the ElGamal encryption scheme, the group $\mathbb{G}$ is both the plaintext and the ciphertext space. The setup, encryption, and decryption functions are defined as follows. Note that we use multiplicative notation for the group operation in $\mathbb{G}$.

- Setup: Choose α uniformly at random from $\mathbb{Z}_p$. The public key pk is $h = g^\alpha$ and the secret key sk is α.
- $\mathrm{Enc}_{\mathrm{pk}}(m)$: Choose r uniformly at random from $\mathbb{Z}_p$. The encryption of m is the tuple $(g^r, h^r \cdot m)$, where h is the public key.
- $\mathrm{Dec}_{\mathrm{sk}}(c)$: Parse the ciphertext c as the tuple (c_0, c_1). Then, compute $\frac{c_1}{c_0^\alpha}$, where α is the secret key.

Correctness is not hard to see. Given a valid ciphertext $c = (g^r, h^r m) = (c_0, c_1)$ for a message m, $c_0^\alpha = g^{r\alpha} = h^r$, and so $\frac{c_1}{c_0^\alpha} = \frac{h^r m}{h^r} = m$. Security of the scheme follows if we assume the Decisional Diffie-Hellman (DDH) assumption holds in the group $\mathbb{G}$. Roughly speaking, the DDH assumption states that the 4-tuple (g, h, g^r, h^r) is computationally indistinguishable from (g, h, g^r, t) where $g, h,$ and t are drawn uniformly from $\mathbb{G}$ and r is drawn uniformly from $\mathbb{Z}_p$. There are many groups in which the DDH assumption is conjectured to hold, for instance, the subgroup of quadratic residues in $\mathbb{Z}_q^*$ where q is a safe prime ($q = 2p + 1$ for a prime p). See [4] for a more thorough treatment.

Finally, we note that the ElGamal encryption scheme is multiplicatively homomorphic. Suppose we have ciphertexts $c_0 = (g^{r_0}, h^{r_0} m_0)$ and $c_1 = (g^{r_1}, h^{r_1} m_1)$ of two messages m_0 and m_1. Consider taking the element-wise product of c_0 and c_1. We obtain a new tuple c where

$$c = (g^{r_0} \cdot g^{r_1}, h^{r_0} m_0 \cdot h^{r_1} m_1) = (g^{r_0 + r_1}, h^{r_0 + r_1} m_0 m_1).$$

Observe that c is an encryption of the product $m_0 m_1$ with randomness $r_0 + r_1$. Thus, in the vanilla ElGamal encryption scheme presented above, multiplying two ciphertexts together yields a new ciphertext that encrypts the product of the underlying plaintext messages. Note though that the randomness in c is not uniform and independent of the randomness in c_0 and c_1. This can be addressed by *re-randomizing* the ciphertext: compute the element-wise product of c with (g^r, h^r) for a freshly generated randomizer r.

The Holy Grail: Fully Homomorphic Encryption

The ElGamal cryptosystem described in the previous section is homomorphic with respect to a single operation. Over the years, numerous other homomorphic encryption schemes have also been developed. For instance, the basic RSA cryptosystem is also homomorphic with respect to multiplication, and the Paillier cryptosystem [5] is homomorphic with respect to addition. However, in all these cases, the encryption schemes are homomorphic with respect to a *single* algebraic operation. While these simple homomorphic cryptosystems have a wide range of applications in secure voting, private information retrieval, and others, the restriction to a single operation renders them incapable of evaluating general transformations on encrypted data. The natural question that Rivest, Adleman, and Dertouzous [6] posed shortly after the discovery of RSA in 1978, was whether there existed an encryption scheme that was *fully homomorphic*, namely, homomorphic with respect to *both* addition and multiplication.

The power of a fully homomorphic encryption scheme (FHE) lies in the fact that it enables arbitrary computation on encrypted data (see Figures 1 and 2 for two simple applications). To see why, suppose we have an encryption scheme that is homomorphic with respect to both addition and multiplication over the finite field $\mathbb{F}_2$. If we view the elements of $\mathbb{F}_2$ as bits, then addition in $\mathbb{F}_2$ is equivalent to taking the "xor" of the input bits or values. Similarly, multiplication in $\mathbb{F}_2$ corresponds to evaluating the "and" of the input bits. Thus, if an encryption scheme is homomorphic with respect to addition and multiplication in $\mathbb{F}_2$ and we view our ciphertexts as encryptions of bits, then our homomorphic operations enable the evaluation of "and" and "xor" gates over the input bits. Since "and" and "xor" gates are universal for the class of Boolean circuits, this means that using only the homomorphic operations, we can evaluate an *arbitrary* Boolean circuit over the encrypted input bits. Thus, a fully homomorphic encryption scheme enables arbitrary computation on encrypted data.

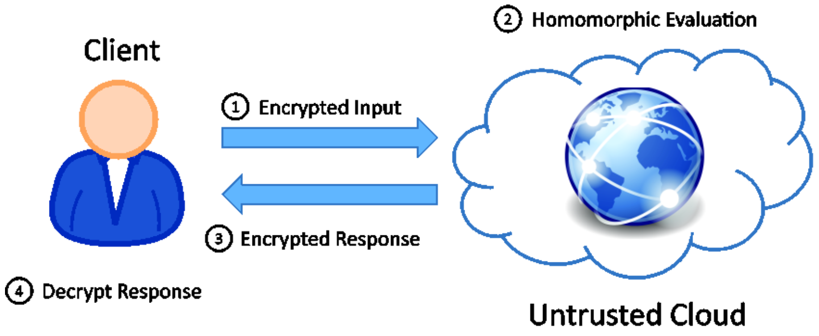

Figure 1. A natural application of fully homomorphic encryption is outsourcing computation to an untrusted third party, such as the cloud. Such a scenario might arise when a client lacks the computational resources to carry out the computation herself, and thus, needs to delegate the computation to a potentially untrustworthy party. In the simplest example of outsourcing computation using FHE, the client first encrypts her input using the FHE scheme; then she sends the ciphertexts to the cloud, who performs the computation homomorphically. Finally, the client receives the encrypted response, and decrypts to learn the result of the computation.

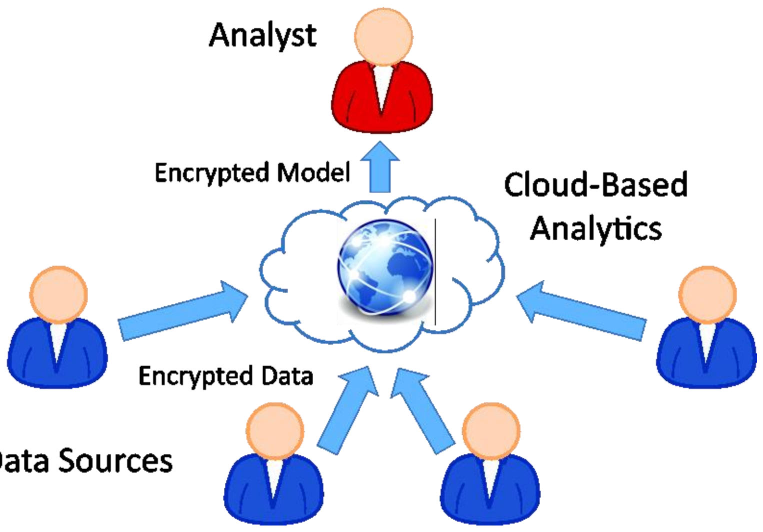

Figure 2. A second application of FHE pertains to cloud-based data analytics. In this example, the goal is to learn a model or compute a function on data collected by several parties. For concreteness, suppose a health specialist is tracking the spread of an epidemic and is requesting patient information from different hospitals. Using FHE, each of the hospitals can submit their records encrypted under the analyst's public key to a cloud-based computing platform. Using the homomorphic properties of the encryption scheme, the cloud then computes some aggregate statistics or develops a model on the input data. The encrypted result of the computation is then provided to the health specialist or analyst. As long as the cloud and the analyst do not collude, the analyst only obtains the model and does not learn about the individual patient records.

Note that it suffices to have homomorphism with respect to addition and multiplication over any finite field $\mathbb{F}_p$ since we can simulate the homomorphic operations over $\mathbb{F}_2$ using the operations over $\mathbb{F}_p$. Finally, as a technical point, we restrict our attention to *compact* homomorphic encryption schemes, that is, schemes in which the size of the ciphertext does *not* grow in the size of the circuit being evaluated. Without this restriction, there are trivial constructions that can be made to satisfy the required properties. For instance, a non-compact (and uninteresting) "fully homomorphic encryption scheme" would be to append the description of the circuit to be evaluated to each ciphertext and delegate the actual computation to the decryption algorithm.

For 30 years after Rivest, Adleman, and Dertouzos first presented the challenge of constructing a fully homomorphic encryption scheme, the problem remained unsolved. Due to both the apparent difficulty of the problem, as well as the tantalizing power afforded by a fully homomorphic encryption scheme, it was considered by some to be the "holy grail" in cryptography. The first major breakthrough in this area came in 2005, with the development of the Boneh-Goh-Nissim (BGN) pairings-based cryptosystem [7]. The caveat, however, was that while the BGN cryptosystem could support an arbitrary number of additions, it could only support a single multiplication. In other words, with the BGN cryptosystem, it became possible to evaluate quadratic functions on encrypted data. The power of the BGN cryptosystem was somewhere in between the simple homomorphic encryption schemes that could evaluate just a single operation and the complete versatility of a fully homomorphic encryption scheme. As such, it was regarded as a *somewhat homomorphic encryption scheme* (SWHE).

Gentry's Blueprint: Bootstrapping for FHE

More than 30 years after the idea of FHE was first described, Craig Gentry presented the first construction of a fully homomorphic encryption scheme in his breakthrough work in 2009 [8]. Gentry's original construction can be broken down into two key ingredients: a somewhat homomorphic encryption scheme that can support a limited number of operations (a few multiplications and many additions), and a bootstrapping transformation that produces a FHE scheme from the SWHE scheme. Gentry's construction has generated tremendous interest within the cryptography community, and in the ensuing years, numerous new schemes that are simpler and more efficient compared to Gentry's original construction have emerged. However, to date, all of these schemes follow Gentry's blueprint of first constructing a SWHE scheme, and then applying the bootstrapping transformation to obtain FHE.

Gentry's bootstrapping transformation is an ingenious method of taking a SWHE scheme and converting it into a FHE scheme. To bootstrap, we begin with a SWHE encryption scheme that supports a limited number of operations. Moreover, suppose that the decryption algorithm for the SWHE scheme can be expressed as a small circuit that can be homomorphically evaluated by the SWHE scheme. In other words, we require that the SWHE scheme is able to homomorphically evaluate its own decryption circuit. Then, we define a $\mathrm{Recrypt}$ function (Figure 3) as follows. Suppose $c = \mathrm{Enc}_{\mathrm{pk}}(m)$ is a ciphertext encrypting the message m under the public key pk, and moreover, suppose we have an encryption of the bits of the secret key sk under the same public key; denote this $\mathrm{Enc}_{\mathrm{pk}}(\mathrm{sk})$.

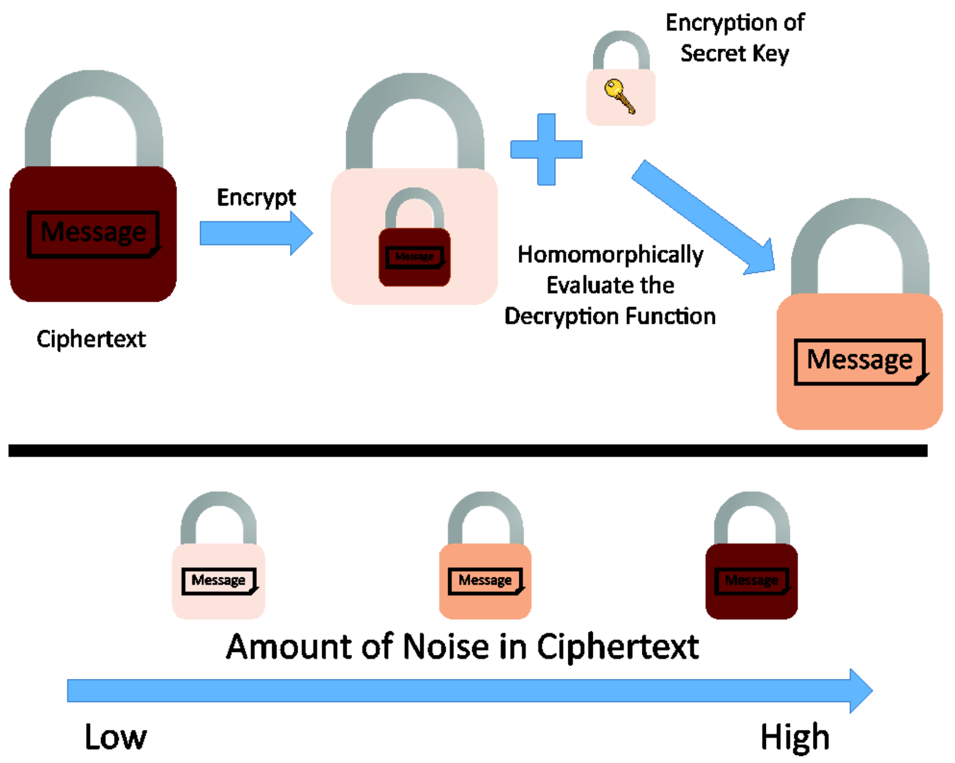

Figure 3. A graphical representation of the Recrypt functionality. All bootstrappable SWHE schemes are based on lattice-based methods where ciphertexts are "noisy." In the Recrypt function, a noisy ciphertext is refreshed by first re-encrypting it to produce a fresh ciphertext (with low noise), and then homomorphically evaluating the decryption function. This yields a ciphertext that encrypts the same message, but with less noise (assuming the decryption function is sufficiently simple).

The Recrypt function then encrypts c under pk to obtain $\mathrm{Enc}_{\mathrm{pk}}(c)$, and homomorphically evaluates the decryption circuit on input $\mathrm{Enc}_{\mathrm{pk}}(c)$ and $\mathrm{Enc}_{\mathrm{pk}}(\mathrm{sk})$. Since $\mathrm{Enc}_{\mathrm{pk}}(c)$ is a *fresh* ciphertext, we can perform as many homomorphic operations on $\mathrm{Enc}_{\mathrm{pk}}(c)$ as the SWHE scheme can provide. Additionally, recall that when we homomorphically evaluate a function f on a ciphertext $\mathrm{Enc}_{\mathrm{pk}}(m)$, we obtain a new ciphertext $\mathrm{Enc}_{\mathrm{pk}}(f(m))$. In this case, the function f is the decryption function $\mathrm{Dec}_{\mathrm{sk}}$. Thus, when we homomorphically evaluate the decryption function, we obtain the ciphertext $\mathrm{Enc}_{\mathrm{pk}}(\mathrm{Dec}_{\mathrm{sk}}(c)) = \mathrm{Enc}_{\mathrm{pk}}(m)$. But this is precisely what we started with! The difference, however, is that if the SWHE scheme has enough homomorphism to support the decryption function and one additional operation, then we can perform one additional homomorphic operation on this new ciphertext, and repeat the procedure *ad infinitum*. Each time we apply the Recrypt procedure, we obtain a ciphertext encrypting the same message, but able to support at least one additional operation. In this sense, we are able to bootstrap the limited homomorphic capabilities of the original scheme to obtain a scheme that is fully homomorphic. One caveat is that we now require the SWHE scheme to be circularly secure, that is, the scheme remains secure even when the adversary is given an encryption of the scheme's secret key.

All known constructions of bootstrappable SWHE schemes are lattice-based. To provide some intuition of what these schemes look like, we sketch out the high-level details of the GSW construction due to Gentry, Sahai, and Waters from 2013 [9]. The GSW scheme fixes a modulus q and a dimension n. The secret key is a vector $v \in \mathbb{Z}_q^n$ with at least one "big" component v_i. The encryption of a message $m \in \mathbb{Z}_q$ in the GSW scheme is an $n \times n$ matrix C over $\mathbb{Z}_q$ (the integers modulo q), such that $Cv = mv + e$, where $e \in \mathbb{Z}_q^n$ is a "small" noise vector (the value of each component of e is small compared to the modulus q). To decrypt a ciphertext, we take the i^{th} row of C (denoted C_i) and compute the quantity $\lfloor \langle C_i, v\rangle / v_i \rceil$, where $\lfloor \cdot \rceil$ denotes the rounding operation. In other words, we evaluate the quotient $\langle C_i, v\rangle / v_i$ over the rationals, and then round the result to the nearest integer. To see that this works, we use the fact that $\langle C_i, v\rangle = mv_i + e_i$ and so

$$\frac{\langle C_i, v\rangle}{v_i} = \frac{mv_i + e_i}{v_i} = m + \frac{e_i}{v_i}.$$

As long as the errors e_i are small compared to v_i (it suffices that $|e_i/v_i| < 1/2$), the decryption function recovers the correct value m.

At a high level, the ciphertexts in the GSW scheme consist of matrices with the following property: the secret key is an *approximate* eigenvector whose associated eigenvalue is the message. It is critical that the secret key is only an *approximate* eigenvector. Without the noise terms, it becomes trivial to break the cryptosystem: given a ciphertext C, we can compute its eigenvalues, and correspondingly, the underlying message, in polynomial time. Adding a small amount of noise allows security to be based on the now-standard *learning with errors* (LWE) problem [10] from lattice-based cryptography.

Without getting into the more technical details of the GSW cryptosystem, we sketch its homomorphic properties. Suppose two ciphertexts C_0 and C_1 are encryptions of messages m_0 and m_1, respectively. Then, the sum $C_0 + C_1$ of the two ciphertexts is an encryption of $m_0 + m_1$. This can be seen through the following simple calculation:

$$(C_0 + C_1)v = C_0 v + C_1 v = m_0 v + e_0 + m_1 v + e_1 = (m_0 + m_1)v + (e_0 + e_1).$$

Thus, $C_0 + C_1$ is a ciphertext encrypting $m_0 + m_1$. Note also that the noise in the new ciphertext has also increased; it is the sum of the noise in C_0 and C_1. Similarly, we can verify that the product C_0C_1 of the two ciphertexts is an encryption of the product m_0m_1:

$$(C_0C_1)v = C_0(m_1v + e_1) = m_0m_1v + m_1e_0 + C_0e_1.$$

Thus, C_0C_1 is a valid encryption of m_0m_1 provided that the new noise term $m_1e_0 + C_0e_1$ is still sufficiently small. As presented so far, there is no reason to believe that this is small (in fact, both m_1e_0 and C_0e_1 can be large if m_1 is large or if C_0 contains large entries). Thus, a few additional tricks are needed to ensure the noise does not blow up with each multiplication. Nonetheless, the basic principles still apply: homomorphic addition corresponds to matrix addition and homomorphic multiplication corresponds to matrix multiplication in the GSW scheme. As noted earlier, the noise in the ciphertexts increases with each homomorphic operation; as a result, the scheme is only somewhat homomorphic. After sufficiently many homomorphic operations, the noise will no longer be small compared to v_i, in which case decryption no longer produces the correct result (the signal has effectively been washed away by the noise). It turns out, however, that the scheme does support enough homomorphism to evaluate its own decryption function, and thus, can be bootstrapped to obtain a fully homomorphic encryption scheme!

Concluding Remarks

The idea of fully homomorphic encryption is almost as old as the concept of public-key encryption. But unlike public-key encryption, the first construction of FHE eluded cryptographers' best efforts for thirty years. Because of the difficulty in attaining FHE and its potential as a primitive for constructing and simplifying other cryptographic protocols, as well as its natural applicability to outsourcing computation, some have come to regard FHE as the holy grail of cryptography. Thus, with Gentry's breakthrough construction in 2009, cryptographers have effectively attained the holy grail; however, Gentry's work does not represent an end to the quest for the holy grail. Rather, his work has ignited a long stream of subsequent research in FHE.

On the theoretical side, a substantial amount of work has focused on developing asymptotically more efficient schemes, schemes based on simpler, or better-understood assumptions, as well as FHE schemes with additional properties. Equally important has been the work on the practical side of implementing fully homomorphic encryption. On the positive side, the performance of bootstrappable SWHE schemes have increased by several orders of magnitude since the earliest implementations of 2009. Nonetheless, there is still considerable overhead in terms of both computational performance as well as parameter sizes in existing implementations that severely limit the practicality and applicability of current implementations. It remains an open and important problem to develop SWHE and FHE schemes that are practical for deployment in our modern, cloud-centric computing environment.

Acknowledgments

The author is currently supported by an NSF Graduate Research Fellowship under grant number DGE-114747.

References

[1] Goldwasser, S. and Micali, S. Probabilistic encryption. *Journal of Computer and System Sciences* 28, 2 (1984), 270-299.

[2] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84*. G. Blakley and D. Chaum (Eds.). Springer, Berlin Heidelberg, 1985, 10-18.

[3] Rivest, R., Shamir, A. and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 2 (1978), 120-126.

[4] Boneh, D. The decision Diffie-Hellman problem. In *Algorithmic Number Theory, Proceedings of the Third International Symposium* (ANTS-III) (Portland, June 21-25). J.Buhler (Ed.). Springer, Berlin Heidelberg, 1998, 48-63.

[5] Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, 1999.

[6] Rivest, R., Adleman, L. and Dertouzos, M. On data banks and privacy homomorphisms. In *Foundations of Secure Computation* 4, 11 (1978), 169-180.

[7] Boneh, D., Goh, E.-J. and Nissim, K. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography, Proceedings of the Second Theory of Cryptography Conference (TCC)* (Cambridge, February 10-12). J. Kilian (Ed.). Springer, Berlin Heidelberg, 2005, 325-341.

[8] Gentry, C. A fully homomorphic encryption scheme. *Doctoral Dissertation, Stanford University*, 2009.

[9] Gentry, C., Sahai, A. and Waters, B. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology, Proceedings of CRYPTO '13*. R. Canetti and J. Garay (Eds.). Springer, Berlin Heidelberg, 2013, 75-92.

[10] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing* (STOC '05). ACM, New York, NY, USA, 2005, 84-93.