Cloud Computing and Homomorphic Encryption

By: Vicki Holzknecht

East Carolina University

Contact: holzknechtvi07@students.ecu.edu

Abstract

Cloud Computing has recently gained quite a bit of momentum throughout the technology community. The Cloud is intertwined in everyday living via mobility platforms, social networking, email, accessing cloud based applications, the list goes on for miles. Cloud Computing allows for services to be delivered by simply having an Internet connection. When encrypted data is sent to the cloud is it truly secure? How safe is the Cloud? The research within this paper will explain how a form of encryption known as homomorphic can be interlaced into the software for truly encrypting data without it ever being decrypted when stored within the cloud.

Keywords: cloud computing, homomorphic encryption

Introduction

Homomorphic encryption is not a new concept for the cryptographic field, the properties have been around since 1978 after the development of the Rivest, Shamir, and Adleman asymmetric algorithm known as RSA. Thirty-one years later in 2009 an IBM researcher Craig Gentry submitted his doctoral thesis on a cryptosystem that combines properties of previous homomorphic algorithms, thus birthing the notion of a fully homomorphic encryption scheme using lattice based cryptography. The remainder of the research is broken down into the following sections: History of Computing, Cloud Computing, Risk Associations of Cloud Computing, Cloud Computing Data Encryption Algorithms and lastly a synopsis on Fully Homomorphic Encryption.

History of Computing

Early existences of computing predate to the fourteenth century usage of a numerical device known as the abacus. The abacus is a simple instrument for performing calculations by counting small beads along rods (Merriam-Webster, n.d.). Travelling through time to the Industrial Revolution, a nineteenth century English mathematician Charles Babbage invented the first automatic adding calculator known as the Difference Engine (Freiberger, Swaine, n.d). Improving upon the concept of the Difference Engine, Babbage wanted to create a prototype of a mechanical digital machine that would perform multiple calculations, designed with four components: the reader, mill, the store and the printer (Freiberger, Swaine, n.d) known as the Analytical Engine.

Even though Babbage never fully completed the Analytical Engine (multiple operational errors), his revolutionary idea of a machine performing any calculation and that can change the instructions of a punch card giving him the name some know him as the “Father of Computing.”

Many of today's computers have similar characteristics as Babbage's Analytical Engine: please refer to Table 1 for a brief comparison of the four components compared to present day computers:

Component Comparison	
Babbage Time	**Present Time**
The Reader (using a punch card)	Keyboard (Input)
Mill	Central Processing Unit (CPU)
The Printer (looking at holes in the punch card)	Graphical Display, Printer (Output)
The Store	Memory and Storage Areas

Table 1 Component Comparison (Freiberger, Swaine, n.d & History-Computer, n.d.)

Cloud Computing

Definitions for computing have changed over the course of time; previously, computing was associated with people (humans) carrying out mathematical calculations, today computer/machines are responsible for performing the mathematical calculations. Today there are many computing architectures: quantum, utility, mobile, grid, cloud and many more; however, this research paper focuses on cloud computing and Craig Gentry's homomorphic encryption scheme that he proposed to address security concerns associated with the cloud.

The National Standard of Institute defines cloud computing as a configurable computing pool of shared resources (servers, applications, services, storage, and networks) that can be on-demand self-service with very little management interaction (Mell, Grance, 2011). Cloud Computing adoption has gained momentum through the years, more businesses are shifting towards cloud based companies, simply for the flexibility of rapid deployment (pay per use), minimal maintenance on hardware, software, network, and businesses are reducing their capital expenditures (*Kaur, Kinger, 2014*).

There are three types of service models that cloud companies provide: Software as a Service (Saas), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Software as a Service, provides applications that are typically web based can be downloaded on end-user's machine. Examples are Office 365, Google Apps and Adobe Cloud. Platform as a Service provides tools and environment allowing for developers to build, test and deploy applications without having to manage the infrastructure of maintaining the application (*Galav, Ghosh, Shrivastav, 2015*). Examples can be Microsoft Azure and Google App Engine. Lastly, Infrastructure as a Service, is an on-demand service providing virtualized access to servers, storage, operating systems, and networking. An example is Amazon Web Services. _Figure 1 scales out the approximate range and control for the cloud user and the cloud provider for each cloud service models.

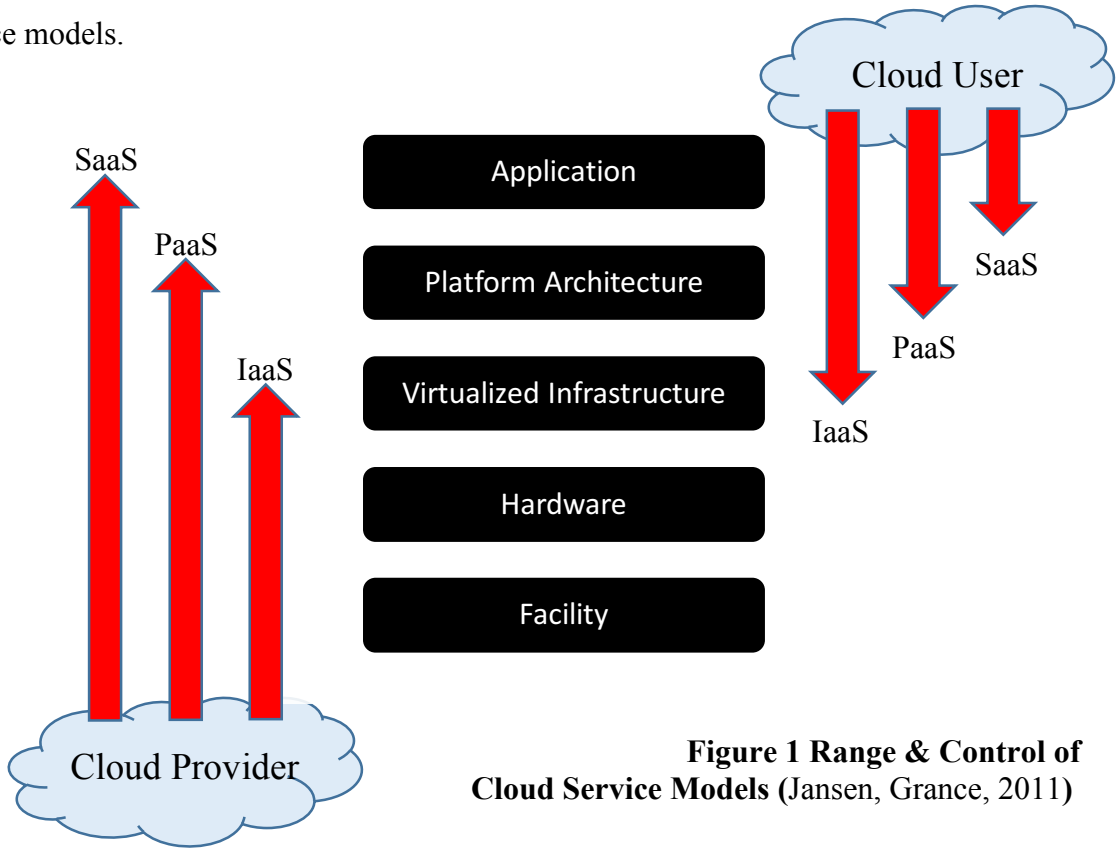

Figure 1 Range & Control of Cloud Service Models (Jansen, Grance, 2011)

Risk Associations of Cloud Computing

Availability, Integrity, and Confidentiality (AIC) are three crucial components of security that is designed to help safeguard the flow, the storage and utilization of information. As businesses drift towards migrating data to the cloud, AIC is threatened. Industry experts and top corporate decision makers made it crystal clear during a Secure Cloud Conference on the focal points to mold the future of cloud computing security: legality matters, accurate reporting of incidents, improving on encryption methods, protecting critical information infrastructures from cyber-attacks and being compliant (*Galav, Ghosh, Shrivastav, 2015*) on security framework programs and standards that assess risk and implement proper security controls.

When information is at risk, who is responsible (cloud provider or cloud user) for maintaining security controls when cloud computing service models has been added to the equation? The answer is both, depending on the model of choice. PaaS the cloud provider is accountable for preserving the integrity and availability of data while cloud user is dealt the cards of privacy control and confidentiality (Zhao, Li, Liu, 2014). SaaS, the cloud provider is burdened for providing security provisions to cover all areas of the AIC (Zhao, Li, Liu, 2014), since the cloud user is not responsible for maintenance on infrastructure (Jansen, Grance, 2011***).*** Lastly, IaaS the cloud provider will assume the responsibility of ensuring data is continuously available, while the cloud user will undertake all other security provisions that go beyond the basic infrastructure of the cloud provider (Zhao, Li, Liu, 2014**)**. Security controls ought to be practiced within a business/corporate environment before a cloud service model has been chosen.

The Cloud Security Alliance (CSA) is a non-profit organization who promotes cloud computing standards for a secure environment (Thomas, Jose, Afsar, 2013). CSA conducted a

survey amongst industry experts to establish top security concerns within cloud computing. The treacherous twelve are ranked by severity from the survey:

1) Data Breaches
2) Weak Credential Management
3) Vulnerable Application Program Interfaces (APIs)
4) System and Application Exposures
5) Account Hijacking
6) Vengeful Insiders
7) Advanced Persistent Threats
8) Data Loss
9) Insufficient Due Diligence
10) Abusing Cloud Services
11) Denial of Service
12) Shared Technology Issues

(Brooks, Field, etal, 2016)

Cloud Computing Data Encryption Algorithms

What is encryption and decryption? Encryption is the process of using cryptographic algorithms to jumble the contents of a video, text, image, etc, turning the data into an indecipherable format (Nigoti, Jhuria, Singh, 2013) preventing unauthorized use. Decryption is the reverse process of encryption, typically the user who originally encrypted the data will need to have a private key to unlock or restore the original data. There are basically two types of encryption algorithms: symmetric and asymmetric. The Symmetric algorithm both of the users (sender / receiver) who are going to establish a secret communication session with one another must agree on one key to encrypt and decrypt the data (Ahila, Shunmuganathan, 2014). Asymmetric algorithm during this secret communication both the sender and the receiver have their own public and private keys. The Public key is used to encrypt the data and a Private key is what is used for decrypting the data (Nigoti, Jhuria, Singh, 2013). From a cloud computing perspective, databases can bloat to a pretty decent size, if encrypted using the asymmetric algorithm the performance (decrypting the database) is much slower than a database that has been encrypted using a symmetric algorithm (Jasim, Abbas, El-Horbaty, Salem, 2013).

The most widely used asymmetric encryption algorithm is the Ron Rivest, Adi Shamir and Leonard Adleman 1978 cryptosystem RSA, named after the developers (Kaur, Kinger, 2014). RSA was one of the first partially homomorphic schemes that used multiplication property (Benzekki, Fergougui, Alaoui, 2016). At the time it was the most practical public-key cryptosystem and by adding random padding to the encryption process, provided a layer of security (Yi, Paulet, Bertino, 2014). Without getting bogged down into the mathematical formulas of RSA; note that RSA uses a public key that can be known by everyone to encrypt the raw data and generate a different "secret" private key to decrypt data; essentially plaintext is being transformed into ciphertext.

In 1999 Pascal Paillier invented the Paillier encryption scheme uses addition property (Benzekki, Fergougui, Alaoui, 2016), instead of the multiplication property, however it does support the multiplication but two key generators would need to be made, one for RSA (multiplication) and the other for Paillier (addition) (Tebaa, Haji, 2013). Paillier uses a method based on residue classes that is computationally difficult (Yi, Paulet, Bertino, 2014). To encrypt the data, a random integer is chosen using the modulus of *n* feature, meaning that a plaintext message could have multiple ciphertext after encryption has completed (Ahila, Shunmuganathan, 2014). On a side note Paillier has been suggested for use in voting systems, and watermarking (Singh, Dutta, 2014).

Advanced Encryption Standard is a symmetric block cipher encryption developed by the National Institute of Standards and Technology (NIST, 2001) in 2001 for Federal agencies to encrypt unclassified information using block keys of 128, 192, and 256 bits and blocks of 128 to decrypt (NIST, 2001), it is also invulnerable to most cyber-attacks except brute force (Bradford, 2014). Brute force attacks on AES run an exhaustive program trying every permutation in the

128, 192, and 256 ciphers attempt to crack the decipher. Table 2 is a list of other symmetric and asymmetric encryptions along with the ones that were briefly discussed above.

Existing Algorithms	
Symmetric	**Asymmetric**
Data Encryption Standard (DES)	Rivest, Shamir, Adleman (RSA)
BlowFish	Digital Signature Algorithm (DSA)
RC5	Diffie-Hellman Key Exchange
Triple DES (3DES)	El Gamal
Advanced Encryption Standard (AES)	Paillier

Table 2 Existing Algorithms (Kaur, Kinger, 2014)

There are three categories of homomorphic encryption: Partially Homomorphic (PHE), Somewhat Homomorphic (SWHE) and then Fully Homomorphic encryption (FHE) (Singh, Dutta, 2014). Partially Homomorphic is when the algorithm can only compute one operation on the data either multiplying or addition (Singh, Dutta, 2014). Somewhat Homomorphic supports more than one operation such as multiplying or addition, however there is a limitation on the number of multiplying and addition operations that can occur (Singh, Dutta, 2014). Fully Homomorphic can compute arbitrary numbers of both multiplication and addition simultaneously of plaintext (Armknecht, Boyd, etal. 2014). Unlike the other categories of homomorphic encryption, Fully is never decrypted in the cloud, any calculations on the ciphertext, returns with encrypted results to the user; decryption will match the calculations that were made on the plaintext (Ahila, Shunmuganathan, 2014); the result is the same as if the data had never been encrypted in the first place.

Fully Homomorphic Encryption

Orthodox encryption approaches for the cloud consisted of users encrypting the data before uploading a database, application, etc to the cloud. Making simple queries on search engines, databases, editing files would require that encrypted data to be decrypted while in the

cloud (Bajpai, Srivastava, 2014). See any security weaknesses here? The user would need to provide the cloud provider with the private (secret) key, have them uploaded to the server, then decryption process begins (Tebaa, Hajii, 2013). Confidentiality and privacy of the file possibly could be compromised, cloud provider could use the information for their own agenda such as data mining for advertisements (Falkenrath, Rosenzweig, 2012).

Definition of homomorphic encryption: if Enc(a) and Enc(b), etc allowed any arbitrary function *f* to perform computations on Enc(f(a,b)) where *f* can be computing addition or multiplication by not having to use the private key (*Galav, Ghosh, Shrivastav, 2015).*

Craig Gentry's fully homomorphic scheme consisted of an effective algorithm KeyGen, Encrypt, Evaluate and Decrypt (Gentry, 2009). The KeyGen generates a key pair: secret key (sk) and a public key (pk). Encrypt will take the public key and turn the plaintext into ciphertext, $\text{Encrypt}_{\varepsilon}(\text{pk}, \pi_{\iota})$ ε denotes fully homomorphic scheme π_{ι} is just a given ciphertext to encrypt (Gentry, 2009). $\psi \leftarrow \text{Evaluate}_{\varepsilon}(\text{pk}, \text{C}, \ \psi_1 \ldots . \psi_t)$ such that $(\text{Decrypt}_{\varepsilon}(\text{sk}, \psi) = \text{C}(\pi_1 \ldots \pi_{\text{t}})$ (Gentry, 2009). C stands for Arithmetic Circuit, and ψ_{ι} denotes ciphertext. What the above line signifies, when the user queries or makes a changes to the encrypted data, throughout the evaluate stage the secret key is never exposed, and the public key is encrypted (Hayes, 2012). The public key is actually embedded with a "hint" of the secret key, however the hint is not enough to fully decrypt an output of a ciphertext (Gentry, 2009). Decrypt runs within evaluate, where the secret key is given, producing a new encryption of the ciphertext lowering noise this is called refresh (Hayes, 2012), which is part of the bootstrapping technique (Ducas, Micciancio, 2015). Bootstrapping could take at least 30 minutes to run computations if the public key size was 2.3 Gbytes (Beunardeau, Connolly, etal., 2016). Lattice based cryptography was chosen because of the simple arithmetic operations to design for a bootstrappable encryption scheme

(Gentry, 2009). Bootstrapping allows for the evaluation of arbitrary circuits to be decrypted homomorphically (Beunardeau, Connolly, etal., 2016); but because of the complexity of the homomorphic decryption process, fully homomorphic encryption is not currently practical for implementation (Ducas, Micciancio, 2015) at this time.

The results are returned back to the user as encrypted, their secret key is what will be used to decrypt the queried results now residing on the user's local machine.

Ciphertext data contains random numerical noise, with every operation performed, multiplication squares the noise, while addition doubles the noise, (Hayes, 2012) eventually accumulated noise will cause the ciphertext to be undecipherable, so there needs to be a limitation set on how many operations can be computed (Beunardeau, Connolly, etal., 2016). Even though refreshing data sounds useful when controlling the noise, Gentry discovered that the circuit for decryption was too shallow and the evaluate function when running decrypt would gain excessive noise (Hayes, 2012). Squashing would be Gentry's solution, while decrypt is running within evaluate, the circuit size will expand at the expense of increasing the size of the secret key and the ciphertext (Gentry, 2009), allowing for an easier decryption process of the ciphertext (Beunardeau, Connolly, etal., 2016).

The basic perception of fully homomorphic encryption is this: user encrypts data locally before uploading it to the cloud provider. When a query is made on the encrypted data in the cloud, a blind processing of arithmetic computations occurs, allowing for the query results be returned encrypted to the user who made the initial query; the user will use the secret key to decrypt encrypted response (Bajpai, Srivastava, 2014). The cloud provider is unable to view the contents of the encrypted queried data. Decryption never fully occurs inside the cloud, there is

just enough of a "hint" of the secret key embedded within the public key to allow the homomorphic encryption.

Mr. Gentry stated with the extra layer of encryption (refreshing/bootstrapping) the fully homomorphic encryption runs sluggishly for real-world use and within five to ten years after adjusting the scheme, fully homomorphic encryption would be ready to accommodate querying databases, other applications (Naone, 2011), and retrieving results through a search engine, etc. A Google search with Gentry's scheme would take a "trillion times longer" because of the heavy computational requirements of the fully homomorphic encryption (Greenberg, 2014).

Conclusion

Fully Homomorphic Encryption is a start, allowing businesses to maintain control over confidential data (Falkenrath, Rosenzweig, 2012), when the forward jump into the cloud has been decided. Even though this is a groundbreaking accomplishment for cryptographist globally, more research is required before fully homomorphic encryption can go into production for practical usage. Cryptographers since Gentry released his fully homomorphic scheme have either made improvements to his idea, or spawned off other fully homomorphic system. Gentry is continuously working on optimizing his scheme since debut of his 2009 fully homomorphic concept.

References

*Ahila, S. S., & Dr.K.L.Shunmuganathan. (2014). State of art in homomorphic encryption schemes. *International Journal of Engineering Research and Applications, 4*(2), 37-43.

Armknecht, F., Boyd, C., Carr, C., Gjosteen, K., Jaschke, A., Reuter, C., & Strand, M. (2014, December 14). A Guide to Fully Homomorphic Encryption. *International Association for Cryptologic Research,* 1-35. Retrieved from https://eprint.iacr.org/2015/1192.pdf

*Bajpai, S., & Srivastava, P. (2014). A Fully Homomorphic Encryption Implementation on Cloud Computing. *International Journal of Information and Computation Technology, 4*(8), 811-816. Retrieved from http://www.irphouse.com/ijict_spl/ijictv4n8spl_05.pdf

*Benzekki, K., Kamal Benzekki, Abdeslam El Fergougui, & Abdelbaki El Belrhiti El Alaoui. (02/01/2016). International journal of advanced computer science & applications: A secure cloud computing architecture using homomorphic encryption IJACSA.

*Beunardeau, M., Connolly, A., Geraud, R., & Naccache, D. (2016). Fully homomorphic encryption: Computations with a blindfold. *IEEE Security & Privacy, 14*(1), 63-67. doi:10.1109/MSP.2016.8

Bradford, C. (2014, July 31). 5 Common Encryption Algorithms and the Unbreakables of the Future - StorageCraft. Retrieved from http://www.storagecraft.com/blog/5-common-encryption-algorithms

Brooks, J., Field, S., Shackleford, D., Hargrave, V., Jameson, L., & Roza, M. (2016, February). CLOUD SECURITY ALLIANCE The Treacherous 12 - Cloud Computing Top Threats in 2016. Retrieved from https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdf

*Ducas, L., & Micciancio, D. (2015). FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. *Advances in Cryptology -- EUROCRYPT 2015 Lecture Notes in Computer Science,* 617-640. doi:10.1007/978-3-662-46800-5_24

Falkenrath, R., & Rosenzweig, P. (2012, October 5). Op-ed: Encryption, not restriction, is the key to safe cloud computing. Retrieved from http://www.nextgov.com/cloud-computing/2012/10/op-ed-encryption-not-restriction-key-safe-cloud-computing/58608/

Freiberger, P & Swaine, M. (n.d.). Analytical Engine. Retrieved from https://www.britannica.com/technology/Analytical-Engine

*Galav, D. S., Ghosh, S. M., & Shrivastav, P. (2015). Data confidentiality for secure cloud computing through homomorphic encryption. *International Journal of Advanced Research in Computer Science, 6*(1)

*Gentry, C. (2009). *A fully homomorphic encryption scheme.* Retrieved from https://crypto.stanford.edu/craig/craig-thesis.pdf

Greenberg, A. (2014, November 3). Hacker Lexicon: What Is Homomorphic Encryption? Retrieved from https://www.wired.com/2014/11/hacker-lexicon-homomorphic-encryption/

*Hayes, B. (2012). Alice and bob in cipherspace. *American Scientist, 100*(5), 362. Retrieved from http://search.proquest.com.jproxy.lib.ecu.edu/docview/1040716116?accountid=10639

History of Computers and Computing, Babbage, Analytical Engine. (n.d.). Retrieved from http://history-computer.com/Babbage/AnalyticalEngine.html

*Jansen, W. A., & Grance, T. (2011, December). *NIST* (National Institute of Standards and Technology, US Department of Commerce). Retrieved from http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf

*Jasim, O., Abbas, S., El-Horbaty, E., & Salem, A. (2013, December 25). Efficiency of Modern Encryption Algorithms in Cloud Computing. *International Journal of Emerging Trends and Technology in Computer Science, 2*(6). Retrieved from http://www.ijettcs.org/Volume2Issue6/IJETTCS-2013-12-25-095.pdf

*Kaur, R., & Kinger, S. (2014). Analysis of Security Algorithms in Cloud Computing. 3(3), 171-176. Retrieved from http://ijaiem.org/volume3issue3/IJAIEM-2014-03-17-048.pdf

*Mell, P., & Grance, T. (2011, September). *NIST* (National Institute of Standards and Technology, US Department of Commerce). Retrieved from http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Naone, E. (2011, April 19). Homomorphic Encryption - MIT Technology Review. Retrieved July 21, 2016, from http://www2.technologyreview.com/news/423683/homomorphic-encryption/

*Nigoti, R., Jhuria, M., & Singh, S. (2013). A Survey of Cryptographic Algorithms for Cloud Computing. *International Journal of Emerging Technologies in Computational and Applied Sciences,* 141-146. Retrieved from http://iasir.net/IJETCASpapers/IJETCAS13-123.pdf

National Institute of Standards and Technology (NIST). (2001, November 26). Announcing the Advanced Encryption Standard (AES). Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

*Singh, V. K., & Dutta, M. (2014). Secure cloud network using partial homomorphic algorithms. *International Journal of Advanced Research in Computer Science, 5*(5)

*Tebaa, M., & El Hajii, S. (2013). Secure cloud computing through homomorphic encryption. *International Journal of Advancements in Computing Technology, 5*(16), 29-38. Retrieved from http://search.proquest.com.jproxy.lib.ecu.edu/docview/1622028622?accountid=10639

*Thomas, G., Jose V, P., & Afsar, P. (2013). Cloud computing security using encryption technique. Retrieved from http://arxiv.org/abs/1310.8392

Merriam-Webster Dictionary. Abacus. (n.d.). Retrieved from http://www.merriam-webster.com/dictionary/abacus

*Yi, X., Paulet, R., & Bertino, E. (2014). *Homomorphic encryption and applications* (2014th ed.). Cham: Springer Verlag. doi:10.1007/978-3-319-12229-8

*Zhao, F., Li, C., & Liu, C. F. (2014). A cloud computing security solution based on fully homomorphic encryption. Paper presented at the 485-488. doi:10.1109/ICACT.2014.6779008

National Institute of Standards and Technology (NIST). (2001, November 26). Announcing the Advanced Encryption Standard (AES). Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Note: Asterisks denotes journals