network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits. AES operates on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field. The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The numbers of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

High-level description of the algorithm

1. KeyExpansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. InitialRound: AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
3. Rounds

a) SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.

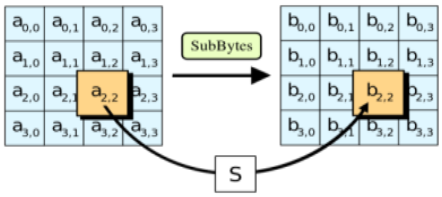

Figure 3: SubBytes module

b) ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps

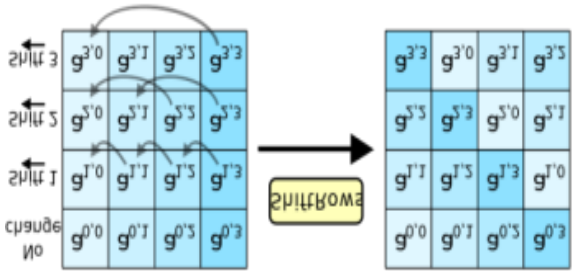

Figure 4: ShiftRow module

c) MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.

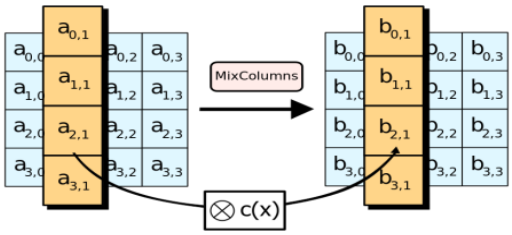

Figure 5: MixColumns module

d) AddRoundKey: In this step, each byte of the state is combined with a byte of the round subkey using the XOR operation

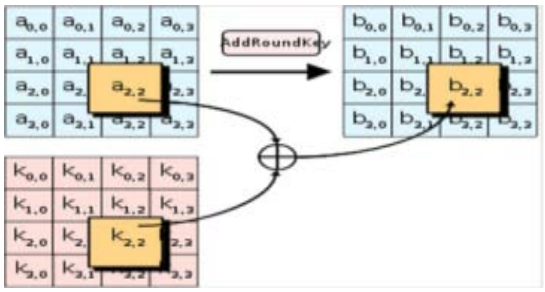

Figure 6: AddRoundKey module

Step 2: Homomorphic Algorithm

Homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This is sometimes a desirable feature in modern communication system architectures. Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services, for example a chain of different services from different companies could 1) calculate the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services.[1] Homomorphic encryption schemes are malleable by design. This enables their use in cloud computing environment for ensuring the confidentiality of processed data. In addition the homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems, collision-resistant hash functions, private information retrieval schemes, and many more.

High-level description of the algorithm:

Definition: Let the message space (M, o) be a finite (semi-)group, and let σ be the security parameter. A *homomorphic public-key encryption scheme* (or *homomorphic cryptosystem*) on M is a quadruple (K, E, D, A) of probabilistic, expected polynomial time algorithms, satisfying the following functionalities:

- **Key Generation:** On input 1^{σ} the algorithm K outputs an encryption/decryption key pair $(k_e, k_d) = k \in K$, where K denotes the key space.
- **Encryption:** On inputs $1^{\sigma}, k_e$ and an element $m \in M$ the encryption algorithm E outputs a ciphertext $c \in C$, where C denotes the ciphertext space.
- **Decryption:** The decryption algorithm D is deterministic. On inputs $1^{\sigma}, k$, and an element $c \in C$ it outputs an element in the message space M so that for all $m \in M$ it holds: if $c = E(1^{\sigma}, k_e, m)$ then $Prob[D(1^{\sigma}, k, c) \neq m]$ is negligible, i.e., it holds that $Prob[D(1^{\sigma}, k, c) \neq m] \leq 2^{-\sigma}$.
- **Homomorphic Property:** A is an algorithm that on inputs $1^{\sigma}, k_e$, and elements $c_1, c_2 \in C$ outputs an element $c_3 \in C$ so that for all $m_1, m_2 \in M$ it holds: if $m_3 = m_1 \, o \, m_2$ and $c_1 = E(1^{\sigma}, k_e, m_1)$, and $c_2 = E(1^{\sigma}, k_e, m_2)$, then $Prob[D(A(1^{\sigma}, k_e, c_1, c_2)) \neq m_3]$ is negligible.

6. Conclusion

In this paper, accessible privacy model and a patient self controllable multi-level privacy preserving cooperative authentication scheme realizing three different levels of security and privacy requirement in the distributed healthcare cloud computing system using AES and Homomorphic encryption is proposed. In future its being a mobile android application we can include GPS to find near by hospitals to help patients with the popular physician based on the number of times the hospitals page is viewed.

References

[1] http://www.cloud-council.org/cscchealthcare110512.pdf
[2] http://www.eweek.com/c/a/Health-Care-IT/Cloud-Computing-in-Health-Care-to-Reach-54-Billion-by-2017-Report-512295
[3] J. Zhou and Z. Cao, *TIS: A Threshold Incentive Scheme for Secure and Reliable Data Forwarding in Vehicular Delay Tolerant Networks*, In IEEE Globecom 2012.
[4] S. Yu, K. Ren and W. Lou, *FDAC: Toward Fine-grained Distributed Data Access Control in Wireless Sensor Networks*, In IEEE Infocom 2009.
[5] F.W. Dillema and S. Lupetti, *Rendezvous-based Access Control for Medical Records in the Pre-hospital Environment*, In HealthNet 2007.
[6] J. Sun, Y. Fang and X. Zhu, *Privacy and Emergency Response in Ehealthcare Leveraging Wireless Body Sensor Networks*, IEEE Wireless Communications, pp. 66-73, February, 2010.
[7] J. Zhou and M. He, *An Improved Distributed Key Management Scheme in Wireless Sensor Networks*, In WISA 2008.
[8] J. Zhou, Z. Cao, X. Dong, X. Lin and A. V. Vasilakos, *Securing m-Healthcare Social Networks: Challenges, Countermeasures and Future Directions*, IEEE Wireless Communications, vol. 20, No. 4, pp. 12-21, 2013.
[9] J. Bethencourt, A. Sahai, and B. Waters, *Ciphertext-Plicy Attribte- Based Encryption*, In IEEE Symposium on Security and Privacy, 2007.
[10] N. Cao, Z. Yang, C. Wang, K. Ren and W. Lou, *Privacy-preserving Query over Encrypted Graph-structured Data in Cloud Computing*, ICDCS'11.
[11] S. Yu, K. Ren and W. Lou, *FDAC: Toward Fine-grained Distributed Data Access Control in Wireless Sensor Networks*, In IEEE Infocom 2009.
[12] F.W. Dillema and S. Lupetti, *Rendezvous-based Access Control for Medical Records in the Pre-hospital Environment*, In HealthNet 2007.
[13] PSMPA: Patient Self-controllable and Multi-level Privacy-preserving Cooperative Authentication in Distributed m-Healthcare Cloud Computing System Jun Zhou, Xiaodong Lin, *Senior Member, IEEE* Xiaolei Dong, Zhenfu Cao, *Senior Member, IEEE*