iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** V **Month of publication:** May 2019

DOI: https://doi.org/10.22214/ijraset.2019.5460

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue V, May 2019- Available at www.ijraset.com

Enriching Secure Secret Data Deletion through Public Variability

Vishal Rasal[1], Prof. Samadhan Sonavane[2]
[1, 2]*Dept. of Computer Science & Engineering, Sandip University, Nashik, India*

Abstract: ***As the volume of the data increases at the user's end, they often outsource the task of data storage to the third parties. Generally, these third parties may be a Cloud Service Provider or Data Repositories or may be small data storage vendors. Each and every data storage service provider offers better options for the end user and lure them to increase their business. This also includes data access, data sharing and much more and Most of the times due to these schemes, encryption keys of the users often compromised to lead to data theft or hacking. So a Data eraser, or key eraser systems are brought into light to solve this, Where the key that is used to decrypt the data is deleted secretly to maintain the integrity of the user data. Some methodologies do exist to achieve this by using the help of the some hardware modules. So it is always a challenging task to achieve the same by providing a software solution. As a tiny step towards this proposed model uses the MAC address to encapsulate the private key and then delete it at the Trusted Platform module end. The proposed model uses the RSA Asymmetric key scheme to generate the private and public keys.***

Keywords: RSA, Public key, Private Key,TPM, Data Audtion.

I. INTRODUCTION

Data security is one of the most essential concepts that are responsible for a safe and secure environment in the computing landscape. Data Security is an act of keeping the data safe from data leakages and unauthorized access. The data security is utilized mainly for the purpose of preservation of the privacy of the data to safeguard it from the hackers and prevent access to unauthorized personnel that could lead to a form of data leaking.

Data can be anything, but data is defined as a form of information in its raw format. It is usually stored in relational databases, personal computers, and network servers, etc. There is a lot of information that is being interpreted as data which can be owned by any person. Most of the personal data of individuals are deemed top secret to that individual and should be maintained that way . Most of the information that is being processed and stored, do not leave the resident machine due to its private nature. And any access to this data which isn't authorized could lead to a lot of problems for the user or the organization that is affected by the breach. It can be your personal data such as credit card details and other personal data, it could also be intellectual property or market analytics data belonging to the organization, which could be worth a lot of money.

This type of information is not supposed to be left without any security and is most of the time inside the target PC and never leaves from there. This is due to the fact that it needs to be safeguarded and multiple locations would ensure multiple weak points where the data could be breached. Therefore, any unauthorized entry and data manipulation would be highly detrimental to the organization and also to the person if the personal data has been manipulated.

Recently there has been a lot of growing interest in this field of Data Security, due to the emergence of the Internet. There has been a lot of uncertainty and as the devices are connected to the internet, there is a big chance an intruder might be able to get their hands on the machine through the internet. The platform has its own drawbacks and conveniences which can be useful or deadly in the hands of an individual. One of the most common forms of protection of the data is the use of encryption. Encryption is one of the most common and widely used mechanisms for ensuring the safety and privacy of the data. Encryption is highly useful in this internet era of wide networks and prevalent personal computing machines alike. Encryption is one of the most essential forms of immediate data security as it involves the scrambling of the data according to various mathematical calculations and algorithms.

The scrambling of the data makes it unreadable or undecipherable. If due to some negligence or a breach, the data is compromised and lands in the hands of the intruder, it would be useless without a key as it would be highly scrambled. This is very useful and is used to safeguard important data efficiently. This type of preventive measures still maintains the privacy of the data even in the scenario of a breach.

The RSA algorithm which stands for the Rivest, Shamir, and Adleman, who are the three pioneers from Massachusetts Institute of Technology who developed this algorithm in the year 1977. It was one of the first algorithms to be publicly elaborated. It now forms

a basis of a cryptosystem which encompasses the various algorithms that are utilized to protect and preserve private and sensitive data.

This technique called RSA utilizes asymmetric cryptography or public key cryptography as it is commonly known. What that means is that the RSA algorithm utilizes two different keys to encrypt the target data. The first key is the public key, which is meant to be shared to the public and the other is intended to be kept private and is called the private key. Both the public key and the private key are linked with each other mathematically.

The Encryption by the RSA algorithm can be achieved by using either of the two keys the public or the private keys. The data is encrypted by either of the two keys needs to be decrypted with the opposite key. The RSA is an extremely popular encryption algorithm due to this feature, and it is also one of the most used asymmetric algorithms, as it maintains the authenticity, integrity, and confidentiality of the data. The RSA algorithm owes its versatility to the fact that it utilizes two keys that are connected to each other in a way that the data can be encrypted and decrypted with either of the two keys. Only when the decryption is done, it should be done with the opposite key, i.e. if the data is encrypted with the public key, it should be decrypted using the private key. If the data is encrypted using the private key, it requires the public key to be decrypted. This is what makes the system extremely versatile and convenient. The Security of the RSA Algorithm can be attributed to its inner workings as described by the research scholars from the Massachusetts Institute of Technology, which states that the resilience of the system is due to the fact that the numbers used to scramble and encrypt the data are generated using the multiplication of two prime numbers. The product of the two primes numbers is quite easy to produce but it cannot be reversed or the large numbers cannot be factorized to extract the prime numbers.

As the factorization of the large number obtained by the multiplication of two prime numbers is next to impossible as the world's most powerful supercomputer would take an unfeasibly large amount of time to compute such a massive computation. This makes the system highly robust and resilient to attacks and other threats to the data.

The key generation in the RSA algorithm is one of the most complex and critical parts of the algorithm as the algorithm decides the two large prime numbers and finds a product of both of them. The resultant number is then used to create two separate keys, the Public Key and then the Private Key. These two numbers are mathematically linked to one another and have demonstrated a certain characteristic of the keys, the key length, which is usually depicted in the form of bits. RSA is one of the most powerful, widely used and a resilient, highly powerful algorithm for the encryption as it is not susceptible to failure and can easily provide robust security for the data with the ingenious use of Prime Numbers and their products.

In this paper, section 2 is dedicated for the literature review of past work ,Section 3 describes the details of the developmental procedure of the model. Section 4 evaluates the results through some experiments and finally section 5 concludes this research article with the traces of the future scope.

II. RELATED WORK

This section of the literature survey eventually reveals some facts based on thoughtful analysis of many authors work as follows.

Michel Abdalla, Mihir Bellare, and Phillip Rogaway[1]presented the security analysis of DHIES. DHIES is a very natural public key encryption scheme. The method follows standard ideas and practice. Intuitively, it is secure. The authors advised that DHIES was a demonstration of hardness attribute of Diffie-Hellman problems which just haven't been made obvious so far. They apprehend some of these hardness properties as formal assumptions and try to establish a relationship among these assumptions. The three new assumptions are the oracle DH assumption (ODH), the hash DH assumption (HDH), and the strong DH assumption (SDH). The ODH and HDH assumptions evaluate the perception in which a hash function H is independent" of the underlying Diffie-Hellman problem. Adrian Antipa, Daniel Brown, Alfred Menezes, Rene Struik, and Scott Vanstone [2] presented invalid-curve attacks on some elliptic curve key formation and public-key encryption protocols. These public key encryption protocols are effectual if the elliptic curve receiver point does not examine that the point reclines on the proper elliptic curve. The simplest and most successful way to prevent these attacks is to check that a received point Q indeed lies on the right elliptic curve. Preferably, the receiver should perform a full validation on Q. The attacks reinforce the importance of performing validation on public keys in protocols where a public key is combined with the receiver’s static private key. Sarah Diesburg, Christopher Meyers, Mark Stanovich, Michael Mitchell, Justin Marshall, Julia Gould, and An-I Andy Wang [3], presented their third version of True Erase. True Erase is a secure deletion framework that is legacy compatible and per file. It can be stand-alone or taint-based secure deletion solutions or as a building block for encryption. They found that the retrofitting security features to the legacy storage data path are more complex.

Initially, they wanted to create a solution that would work with all popular file systems. But they found that the verification problem became much more tractable when working with file systems with proven consistency properties. After struggling to work

against the asynchrony in the data path, they instead associated secure deletion information with the legacy data path flow. They also verified True Erase and its core logic via cases derived from file-system consistency properties and state-space enumeration.

Zachary N. J. Peterson Randal Burns Joe Herring, Adam Stubblefield and Aviel D. Rubin [4] presented the architecture and algorithms for individual file secure deletion. They designed the system to remove data after a compulsory retention period. They also compare two techniques used for secure deletion that uses the secure overwriting and the authenticated encryption combination. They implemented their technique of secure deletion in ext3cow versioning Linux file system and in a reliable device driver.

Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia [5], presented the cryptographic protocols to deal with uncommon aspects of Victorian voting. They characterized the issues, present arrangements and after that analyze their security properties and portray how they attach into other plan choices. Dan Boneh and Richard J. Lipton [6], introduced a system that enables a user to permanently remove a file from the file system and all backup tapes. The ability to revoke backup copies of files is important and may be of interest to many institutions. They applied cryptography in a new way. They used cryptography to remove information rather than secure it. They suggested using the block cipher with longer keys to encrypt the files. They applied triple DES twice to obtain a 224-bit key.

Christian Cachin, Kristiyan Haralambiev, Hsu-Chun Hsiao and Alessandro Sorniotti [7] introduced the encryption based secure deletion technique. They also provide security definition and first model for encryption based secure deletion technique. Their technique used coercive adversary that acquires the master key when the attack happens. They framework their technique of secure deletion from threshold secret sharing and encryption. They generalized all previous existing frameworks for cryptographic secure deletion. They also presented the Linux file system prototype implementation with policy-based secure deletion technique.

Amos Fiat and Adi Shamir [8] described the easy identification and signature schemes to prove authentication of the user message to any other user without sharing public key. They demonstrate that their scheme is more secure against any message attack.

Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy and Henry M. Levy [9] presented a system called Vanish. It is designed to leverage one or more DHTs (Distributed Hash Tables). Vanish causes sensitive data, such as files, emails, or text messages, to irreparable self-destruct, without any movement on the user's part and without any trusted or centralized system. Their measurement and experimental security analysis shed insight into the robustness of their approach to adversarial attacks.

Akli Fundo, Aitenka Hysi and Igli Tafa [10] presented a review on some proposed techniques from different engineers to sanitize data from SSDs. These techniques were based on hard drives because they were successful in there. From the results, the first method had a half success (50:50), the second method was more successful, except some fails, it had and the third method was worthless because it doesn't guarantee anything. So the problem of sanitizing data from SSDs actually is very present nowadays and engineers are trying to find ways to make this more reliable.

Nikolai Joukov, Harry Papaxenopoulos, and Erez Zadok [11] presented the portable data overwriting system that uses the shred, which is an existing user-mode tool. Their system overwrites the data automatically. Their system is simple, portable and continuously overwrite the data in the instance of power failure also. Their system allows users to balance convenience, security, and efficiency. Their system provides a trash bin like functionality in the instance of slow overwriting to further intensify user benefit. They also design patches for Ext3 journaling file system.

Mahesh Kallahalla, Erik Riedel, Ram Swaminathan and Qian Wang Kevin F [12] introduced the cryptographic primitive unusual uses which applied to the secure storage problem with untrusted servers and a need for owner control key dispersal. Their mechanism is utilized as building blocks to plan Plutus, safe, comprehensive and efficient file system. They build a prototype implementation of their plan by integrating it into open AFS and calculating its performance on micro benchmarks.

Jaeheung Lee, Sangho Yi, Junyoung Heo, Hyungbae Park, Sung Y. Shin and Yookun Cho [13] designed a systematic safe deletion plan for flash file systems. Their scheme stores all the file information in encrypted form, and deletes all keys that were utilized to encrypt the file's data, to safely remove the file. Because of flash memory characteristics, log-structured file systems are commonly utilized in flash file systems. There can be numerous keys for a particular file in a log-structured file system. To remove all keys of a specific file with a single erase operation, they designed data structures and algorithms that ensure that all keys of a specific file are saved in the identical block. The proposed system safely removed both the keys and all the metadata of the file.

Byunghee Lee, Kyungho Son, Dongho Won and Seungjoo Kim[14] analyzed the numerous secure deletion techniques for flash memory and review each government agencies advice for removing data safely so that it can never recover. They show that the previously proposed techniques do not reach the government's recommendations and cannot securely delete data. To control these problems, they proposed a technique that can enhance these frailties and prove that the proposed technique performs finer than the previous techniques. The proposed technique is unique and satisfies the government's recommendations to delete data safely for flash memory. Mithun Paul and Ashutosh Saxena [15] presented here a plan for Proof of Erasability (POE) in a distributed

environment. The focus of their work is on a probing engine/destructor which will investigate the environment and based on the rules on the information store, will rag them partially or completely. This prototype can further extend as a service on Cloud. This destructor will systematically modify/update the most significant bit of every data chunk, thus safeguarding that the information is so manipulative that no part of the information will make any sense to whoever gets hold of it later. On analyzing the scheme they found that their proposition can be examined meticulous and actual.

Daniele Perito and Gene Tsudik [16] presented secure erasure, safe code upgrade and remote attestation in the condition of embedded devices. Having examined prior attestation approaches (both hardware- and software-based), they concluded that the former is too expensive, while the latter too uncertain. They then explored a different approach that generalized the attestation problem to remote code upgrade and safe erasure. Their approach, based on Proofs-of-Secure-Erasure depends neither on safe hardware nor on compact timing constraints. Moreover, although not particularly efficient, it is viable, secure and offers some promise for the future. They also evaluate the practicability of the proposed method in the condition of commodity sensors.

Radia Perlman [17] presented three schemes for supporting assured delete; time-based, custom classes, and individual file on demand. These schemes can be combined. In the same file system, some files can be stored with expiration times, others in a class with a custom keys, others that can be deleted individually, on-demand, in an assured manner, and still others that have no assured deletion properties, so that even if deleted, they would be recoverable as long as they still exist on backup media. The schemes can also be nested, e.g., by having a directory encrypted with a custom key, and individual files in that directory also having expiration dates. This is accomplished by encrypting the file key with both the master class key for that directory and with the master class key for the expiration date. Joel Reardon, Srdjan Capkun and David Basin [18]proposed UBIFSec and analyzed it experimentally to ensure that it is efficient, requiring a small, evenly-leveled increase in flash memory wear and little additional computation time. UBIFSecwas effortlessly added to UBIFS. The cryptographic operations are included perfectly in UBIFS's read/write data path. The changes in key state are handled by the existing index of UBIFS's data nodes.

Joel Reardon, David Basin and Srdjan Capkun [19] explored safe deletion in detail. They defined the simple problem of deleting data objects from a physical medium and proved that this problem has numerous complexities and variation. They surveyed lots of related work in detail by organizing the approaches in terms of associate to the physical medium. They systematized the space of adversaries based on classes of ordered capabilities and related the adversaries to real-world examples; they did the same for the classes of environmental assumptions and behavioral properties. Additionally, they examined two common user level approaches—showing the limitations of their interfaces by illustrating the complexity of ensuring secure deletion.

Joel Reardon, Hubert Ritzdorf, David Basin and Srdjan Capkun [20] developed a common method to the design and analysis of secure removal from persistent media. They used graph theory to reason about adversarial knowledge and developed a graph mutation that maintains the properties on adversarial knowledge that allows straightforward provable secure deletion.

Their mutation subsumes the upgrade behavior of all trees like data structures. They designed and implemented a securely-deleting B-Tree based on this mutation. Their analysis showed that the communication and storage overhead is typically negligible and the skeleton tree's caching of B-Tree nodes is very effective.

III. PROPOSED IDEA DESCRIPTION

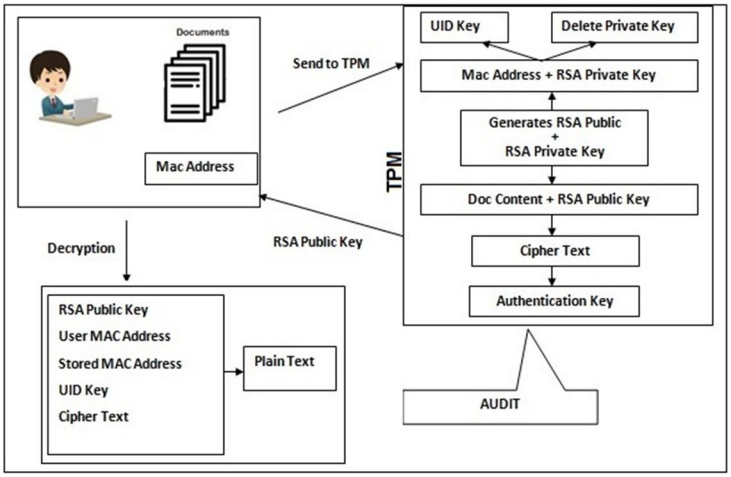


Figure 1: Overview of proposed methodology

1) *Step 1: Data Providing to TPM* - Here in this step user is registered with the developed software at client side. And then login in to the system to upload the plain text file along with the MAC address of the user machine, Which helps the Trusted platform module to identify the user properly.
2) *Step 2 : Key Generation and Encryption* - On receiving the data from the user, TPM then generates the public and private key using the RSA algorithm which are labeled as K_{PUB} and K_{PRI}. Among these two keys K_{PUB} is used to encrypt the plain text using the RSA encryption model. The cipher text is stored properly in a file which is provided to the user on request.
3) *Step 3: Authentication Key generation* - Once the encryption is done, then an authentication key need to generate for the cipher text immediately. This process is carried out using a SHA256 hash key generator which takes the cipher text as input and yields a hash key. Then this hash key is used to generate a 14 byte authentication key called A_{KEY}. The process of generation of A_{KEY} generation can be shown in the algorithm 1.

Algorithm 1: A_{KEY} Generation

// Input : Cipher Text C_T
// Output : Authentication Key A_{KEY}
Function : authenticationKeyGenrator($\mathbf{C_T}$)

a) Step 0: Start
b) Step 1: A_{KEY} =Ø
c) Step 2: HashKey H_K=*SHA256 (*C_T*)*
d) Step 3: N=H_K MOD 7
e) Step 4: *If* N<7, THEN
f) Step 5: P=N+1
g) Step 6: *for* i=0 to A_{KEY} length < 7
h) Step 7: i=i+P
i) Step 8: I*f* i < H_K length, THEN
j) Step 9: $A_{KEY=}$ A_{KEY+} H_K [i]
k) Step 10: H_K =rotate(H_K)
l) Step 11: *End if*
m) Step 12: Else
n) Step 13: i=0
o) Step 14: *End For*
p) Step 15: *End if*
q) Step 16: return A_{KEY}
r) Step 17: Stop

4) *Step 4: Data Audit* - Here data auditing is done to ensure the proper encryption was done correctly or not. For this cipher file is again subjected to create the authentication key called Audit_A_{KEY}. This audit authentication key is matched with the authentication key of the respective file called $A_{KEY.}$ If both the keys are same then it is tagged as properly encrypted.
5) *Step 5: Private key deletion* - This is the major and crucial step of the proposed model, Where the private key K_{PRI} generated by the RSA is going to delete to maintain the highest access level along with the security of the data. This process takes the input of the MAC address belongs to user system along with the K_{PRI}. A hash key is generated from the MAC address using SHA256 scheme which is then used to get the signature for the MAC address of the user using algorithm 1. This signature is referred as U_{ID}. As the U_{ID} is formed then, the original MAC address is used to encapsulate the part of the private key KPRI which was generated through RSA. The characters of the MAC address are fragmented and partial characters of the K_{PRI} are encapsulated in the MAC address which act as the identity of the user. After this K_{PRI} is deleted from the Trusted platform module to ensure the fair and secure transition between the client and service provider.
6) Step 6: Data Decryption - This is the last step of the proposed model, Here user requests for his data, that is stored in the form of cipher text at the TPM's end. To access the data user has to provide the file name to the TPM. And TPM also receives the MAC address of the user machine implicitly to ensure the delivery of the data to the right user. As the TPM receives all the needed attributes from the user then TPM provides the Public key K_{PUB} , Stored signature that is U_{ID} , Stored MAC Address of the user, and Ciphertext $C_{T.}$ The Decryption process is carried out by forming the key for the decryption using the stored MAC

address and the public key. This process is authenticated by the U_{ID} signature and the current MAC ID provided by the user. The complete process of deleting the private key and generating a decryption key is shown in the equations 1,2 and 3.

$MAC_{ST}=MAC + K_{PRI}$ ----------(1)

$K_{PRI} =Ø$ ---------- (2)

$D_{KEY}= MAC_{ST+} K_{PUB}$ ----------(3)

Where,

MAC_{ST} : Stored MAC Address

MAC : User MAC Address

K_{PRI} = Ø: Deleting Private key

D_{KEY} : Decryption Key

IV. RESULT AND DISCUSSIONS

The proposed model of secure secret data deletion system is developed using the Java programming language in windows based laptop. And the Laptop is having the configuration of Corei5 processor along with the 6GB of RAM. The system is completely deployed using the software based interfaces, Which is developed using Netbeans as IDE and Mysql as the Database. The Proposed model is put under the hammer to justify its cause and to measure its efficiency under different circumstances as described below.

When the proposed model is measured for the time taken for the process of Encryption and decryption with the [21] , Which is purely designed the system based on the Java card hardware module. Whereas the proposed model is purely software based which eventually yields the good results compare to [21]. This because of the latency in the transmission between the hardware and software switchovers in [21] is more compare to that of software module transitions.

No of Bytes	Encryption Time by TPM Using Hardware Model (Time in MS)	Encryption Time by TPM Using Secure deletion model (Time in MS)
512	1002	680
1024	1100	688
1536	1050	701
2048	1250	985
2560	2100	1456
3072	2100	1350
3584	2100	1555
4096	2100	1900
4608	3100	2101
5120	3100	2202
5632	3110	2525
6144	3100	2650

Table 1: Encryption Time measurement

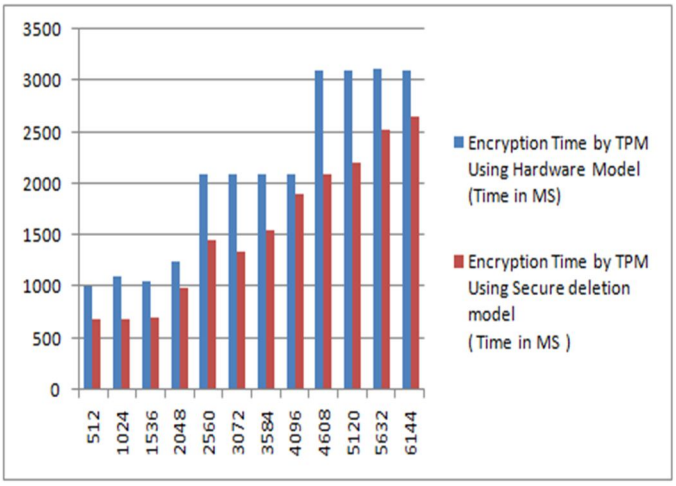


Figure 2: Encryption Time evaluation

No of Bytes	Decryption Time by TPM Using Hardware Model (Time in MS)	Decryption Time by TPM Using Secure deletion model (Time in MS)
512	540	122
1024	550	180
1536	580	245
2048	595	301
2560	750	333
3072	1000	514
3584	980	645
4096	970	670
4608	1500	790
5120	1500	814
5632	1560	908
6144	1600	975

Table 2: Decryption Time measurement

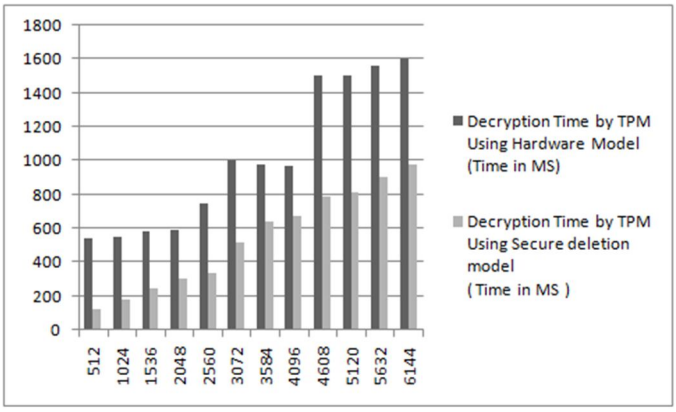


Figure 3: Decryption Time Evaluation

V. CONCLUSION AND FUTURE SCOPE

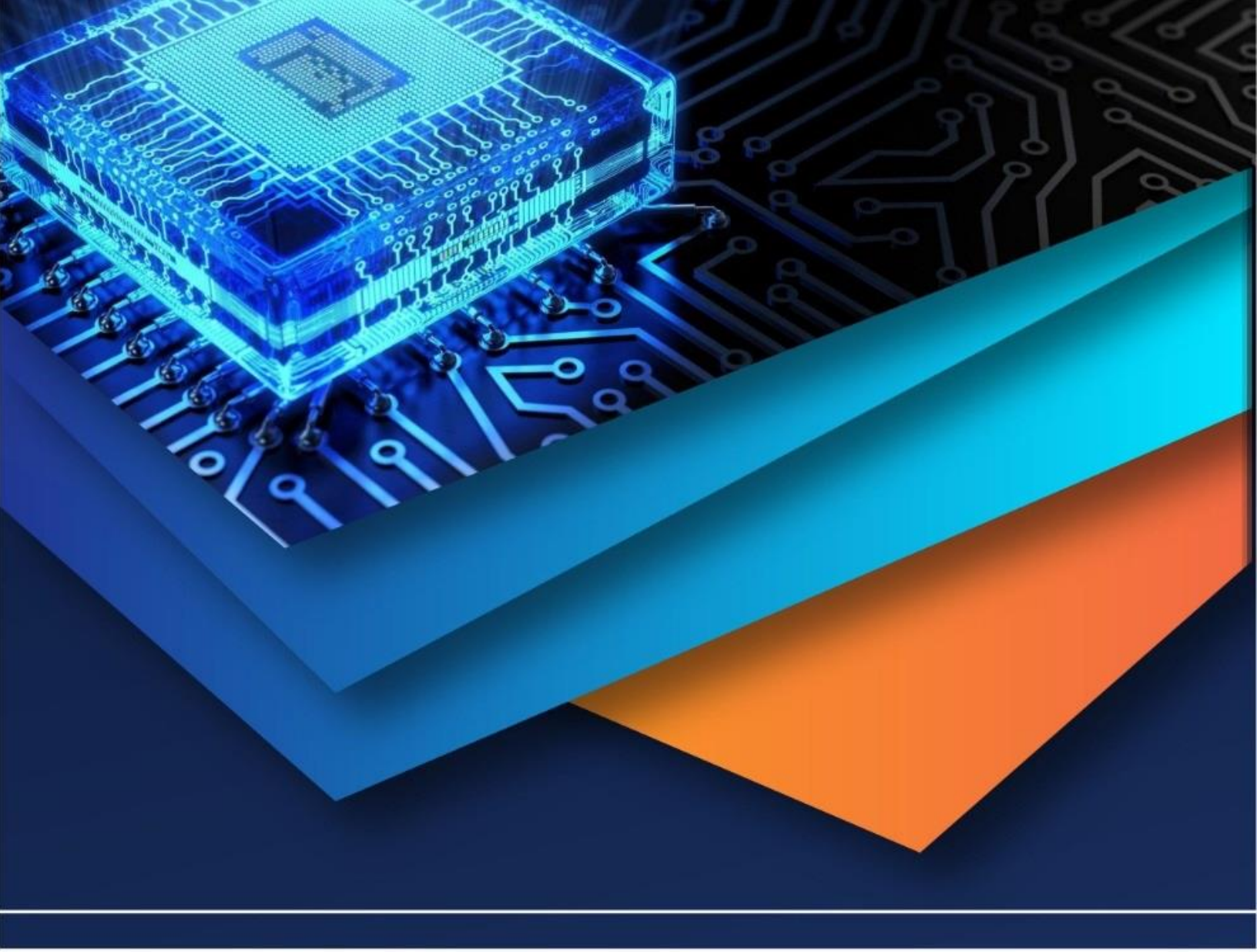
The proposed system of secure deletion of the private keys in the storage system is purely deployed using the software based modules. The proposed model handles the Private key using the MAC address of the user machine to delete the same. And the decrypted files can be obtained by the signature of the MAC address and the Public key of the user. This eventually provides a greater security at the Trusted platform modules and also ensures the high level of accessibility to the users. When the proposed model is compared with that of [21] which is a hardware based model, the proposed system yields better result with respect to the Encryption and Decryption schemes.

In the future the concept of secret data deletion can be applied to heavy data with more secured channelized system in mobile applications too.

REFERENCES

[1] M. Abdalla, M. Bellare and P. Rogaway, "The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES," Topics inCryptology - CT-RSA'01, LNCS Vol. 2020, 2001.
[2] A. Antipa, D. Brown, A., Menezes, R. Struik, S. Vanstone," Validation of Elliptic Curve Public Keys," Proceedings of the6th International Workshop on Practice and Theory in Public Key Cryptography Public Key Cryptography (PKC'03), LNCS2567, pp. 211-223, 2003.
[3] Sarah Diesburg, Christopher Meyers, Mark Stanovich, Michael Mitchell, Justin Marshall, Julia Gould, and An-I Andy Wang, "TrueErase: Per-File Secure Deletion For the Storage Data Path," Research Gate Conference Paper · December 2012, DOI: 10.1145/2420950.2421013.
[4] Zachary N. J. Peterson Randal Burns Joe Herring, Adam Stubblefield and Aviel D. Rubin, "Secure Deletion for a Versioning File System," in Proceedings of the FAST '05: 4th USENIX Conference on File and Storage Technologies.
[5] C, Burton, C. Culnane, J.A. Heather, P.Y.A. Ryan, S. Schneider, T. Srinivasan, V. Teague, R. Wen, Z. Xia, "Using Pret a sVoter in Victorian State Elections," Proceedings of the 2012Electronic Voting Technology/Workshop on Electronic Voting(EVT/WOTE'12), 2012.
[6] D. Boneh, R. Lipton, "A Revocable Backup System," Proceedings6th USENIX Security Conference, pp. 91-96, 1996.
[7] C. Cachin, K. Haralambiev, H.C. Hsiao, A. Sorniotti, "Policy-Based Secure Deletion," Proceedings of the 2013 ACM Conference on Computer and Communications Security (CCS'13), pp. 259-270, 2013.
[8] A. Fiat, A. Shamir, "How to Prove Yourself: Practical Solution to Identification and Signature Problems," Proceedings of CRYPTO, pp. 186-189, 1987.

[9] R. Geambasu, T. Kohno, A. Levy, H.M. Levy, "Vanish: Increasing Data Privacy with Self-Destructing Data," Proceedings of the USENIX Security Symposium, 2009.
[10] Akli Fundo, Aitenka Hysi and Igli Tafa, "Secure Deletion of Data from SSD," (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 5, No.8, 2014.
[11] N. Joukov, H. Papaxenopoulos, E. Zadok, "Secure Deletion Myths, Issues, and Solutions," Proceedings of the second ACM workshop on Storage Security and Survivability (StorageSS),pp. 61-66, 2006.
[12] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, K. Fu," Plutus: Scalable Secure File Sharing on Untrusted Storage," Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST'03), pp. 29-41, 2003.
[13] J. Lee, S. Yi, J.Y. Heo, H. Park, S.Y. Shin and Y.K. Cho,"An Efficient Secure Deletion Scheme for Flash File Systems," Journal of Information Science and Engineering, Vol. 26, pp. 27-38,2010.
[14] B. Lee, K. Son. D. Won, S. Kim, "Secure Data Deletion for USBFlash Memory," Journal of Information Science and Engineering, Vol. 27, pp. 933-952, 2011.
[15] M. Paul, A. Saxena, "Proof Of Erasability for Ensuring Comprehensive Data Deletion in Cloud Computing," Communications in Computer and Information Science, Vol. 89, Part 2, pp.340-348, 2010.
[16] D. Perito, G. Tsudik, "Secure Code Update for Embedded Devices via Proofs of Secure Erasure," Proceedings of the15th European Conference on Research in Computer Security(ESORICS), pp. 643-662, 2010.
[17] R. Perlman, "File System Design with Assured Delete," Proceedings of the Third IEEE International Security in Storage Workshop (SISW), pp. 83-88, 2005.
[18] J. Reardon, S. Capkun, D. Basin, "Data Node Encrypted File System: Efficient Secure Deletion for Flash Memory," Proceedings of the 21st Usenix Symposium on Security, 2012.
[19] J. Reardon, D. Basin, S. Capkun, "SoK: Secure Data Deletion," Proceedings of the 2013 IEEE Symposium on Security and Privacy, pp. 301-315, 2013.
[20] J. Reardon, H. Ritzdorf, D. Basin and S. Capkun, "Secure Data Deletion from Persistent Media," Proceedings of the ACM Conference on Computer and Communications Security, 2013.
[21] Feng Hao, Dylan Clarke, Avelino Francisco Zorzo " Deleting Secret Data with Public Verifiability ", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING,2015.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY