ISSN 2277-3061
Volume 16 Number 3
INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Dynamic Encryption Key Security Scheme (DEKSS) for Mobile and Cloud Systems

Stephen Rodriguez, Paolina Centonze
Iona College 715 North Ave. New Rochelle, New York 10801
srodriguez4@iona.edu
Iona College 715 North Ave. New Rochelle, New York 10801
pcentonze@iona.edu

ABSTRACT

This journal article discusses our Dynamic Encryption Key Security Scheme (DEKSS) and the purpose it serves in providing a new security architecture for protecting databases used in technology stacks involving Mobile and Cloud based devices. Our security scheme is a novel architectural strategy that implements a full-stack architecture for the dispatching and management of data between several Cloud Service Providers (CSP) and any number of mobile devices. This strategy can promise data security needs for both mobile devices and cloud service providers without impacting the security requirements of the other party. While there are limitations in being truly secure, such as those recognized by WhiteHat security in their annual report[1], we believe that our security scheme can effectively circumvent potential threats and secure data through folding data using any number of encryption layers for every table and column of data to be stored. Through this approach, we have found our work to be applicable to a variety of different audiences within the cloud security space.

BACKGROUND

In recent years, the benefits of moving entire software applications to the cloud have become a standard in large and small organizations alike. Although many have seen positive effects from switch to the cloud, there are still many things left to the unknown when we consider the overwhelming number of potential security vulnerabilities when being in the cloud. Most cloud solutions have provided white-paper reports detailing their regulatory actions in securing customer data. These reports focus on assuring the general public about the actions and procedures put in place to provide a secured platform. And yes, despite their ambitious and collective efforts, breaches in their platforms still occur, and have caused even the largest of organizations such as Home Depot, and Yahoo to suffer incredulous losses in revenue and reputation amongst their customers. As a result, these organizations, and many others alike have reached out for new and unique security strategies to protect themselves within the cloud. Some methods, such as encrypting data in the cloud, have become established as a standard practice in the software security space as an adequate method for protection of data in the public domain. Two years ago, Jung Hee Cheon et al. had produced an article on a new hybrid encryption scheme composed from public-key and somewhat homomorphic encryption[2]. In this article, their research emphasized on the idea that it was indeed possible to compose an encryption scheme in which the encrypted data formed could still be used for mathematical calculations, but could still be encrypted in the public space using public-key encryption. In another study published last year by Nguyen-Vu et al. [3], a similar idea was proposed in which selective encryption could be used to receive adequate performance and security. However, this publication also introduced various threat models that could be used in testing the effectiveness of any data security strategy. Although both of these studies

provided favorable results, they both equally fell short in providing adequate details on how to mitigate an active threat attacking their security strategies. Time and time again, encryption schemes such as these can become obsolete due to flawed execution or increased computational power of their adversaries. Instead, what is required is a truly secure methodology that would be able to evolve and reshape itself against a constantly overwhelming adversary whose strength can increase from the many current and future technological advances that can exist. It is a constant race against our own successes and requires us to stay ahead of our competitors. It is for this purpose that DEKSS has been formed into existence and for which it seeks to counter current and future threats to data security.

INTRODUCTION

DEKSS aims to solve data security concerns such as confidentiality, accessibility, and integrity by applying a series of encryption schemes to a set of data in order to reduce the occurrence of large blocks data encrypted with a single encryption key. Similar works such as those performed by Accenture Technology Labs[9], implementing a security strategy that elects multiple encryption schemes rather than a general scheme in order to improve overall data security efforts. Following this premise, we aim to reduce the possibility of large data breaches and pose that it is possible to reduce the number of affected rows in a breach to one single row of data. In comparison to previous related works such as the research article by Hingwe et al.[4], their research focused on introducing a two-layered protection scheme for securing data. Although their research had proven the feasibility of building such a system, it still sustained the potential for leaving an entire database vulnerable to large scale data breaches. In addition, their work fell short of introducing new and unique strategies for Cloud Service Providers to implement in order to contribute to securing their customer's data. As such, our new scheme will address these missing requirements and increase the number of encryption layers to be of any size. For the remainder of the paper, we will discuss topics such as our unique architecture, how to customize the database schema using any number of encrypted layers, how encryption/decryption is performed using multiple schemes, and how to measure your query performance. To begin, we propose that our security scheme provides the following capabilities:

1. Constructs a system to manage the encryption of data

2. Maintains the integrity of plaintext
3. Allows for layering of encryption schemes
4. Efficiently swaps out encryption keys on any layer
5. Secures against current, and future data security threats

DEFINE THREAT MODELS

Before we begin explaining the proposed security scheme, we must be mindful of the following scenarios from our adversaries:

MAN IN THE MIDDLE ATTACK

Man in the Middle is a very common attack model used for testing cryptographic schemes. This model is also known as an eavesdropping attack. In a simple scenario, the attacker is looking at the information that is being exchanged between two entities. An entity can be described as any computing device or one end of a

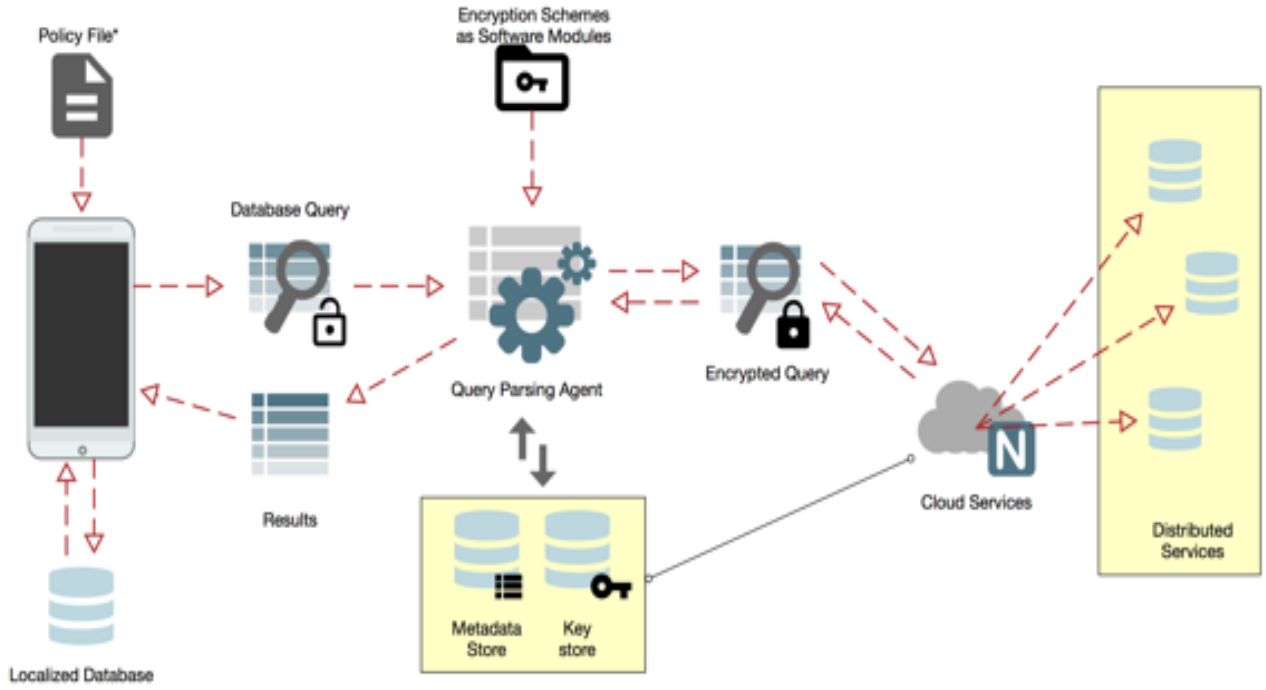
channel of communication. Through this channel, the attacker's goal is to discover and expose a vulnerability through information retrieved within a channel. In our scenario, we will be testing our security strategy to see if it is secure against such an attack. Through testing, this will allow us to confirm that all information being transmitted is secure from prying eyes. Specifically, the only ones able to decipher and read the messages sent will be the sender and the intended receiver.

MALICIOUS DEVICE EXPLOITATION

Protecting against malicious users is a common goal for anyone aiming to protect themselves. However, in this specific scenario, we are using this attack model to define a user who has chosen to exploit some of the configuration options available at the device side. Within our "Data Policy File" section, we will discuss the methods by which we open the security architecture for creating user-defined policies for defining custom methods for securing data. In a way, for example, this is no different than using a guessable password for your email account. As such, we will be using this scenario to provide proofs for more commonly used attack models like Chosen-Cipher-Attack and Chosen-Plaintext-Attack.

THE ARCHITECTURE

In the following sections, we describe the characteristics and components used to develop our proposed security scheme. In addition, we explain the purpose for which each component serves in defining a secure system.



* The Policy File details configuration settings for how the Query Processing Agent encrypt/decrypt queries

THE DATA POLICY FILE

The data policy file is a series of settings for how to manage and access stored data in the cloud. With this

file, a client can provide customized requirements for the interpreting the database schema and for how the security system should handle encrypting queries that will be ran against the cloud database(s). Within these policy settings, we can instruct the security system to know how many layers of encryption to use and in what order. Because of the very nature of this security system, these definitions are required in order for the system to be able to perform any action in safeguarding the original plaintext during transit across each stack. To demonstrate this, let us use an example where we are storing salary numbers in our database, and that we will be performing calculations upon this data through our queries. To begin, the client would write a policy that instructs the security system on how to begin the storage of an encrypted column and table data. The following example JSON structure demonstrates this.

```
{ "tables": {
    "scheme": [ "MD5" ],
    "users": {
      "columns": {
        "salary": { "scheme": [ "RSA" ] }
      }
    ...
```

In this example, we used array to define the order for how the system should encrypt data. Every value corresponds to a unique encryption scheme stored within the query parsing agent stack. It is important for the agent to provide a variety of encryption schemes here in order to be able to encrypt multiple data types. In a recent article published by Mohammad et al. titled “Enhanced Data Security Model for Cloud Computing”[5], securing data is not so much about the strength of an encryption scheme but rather the application of type specific schemes to target data models effectively for maintaining and increasing query performance.

MODELING THE ENCRYPTION LAYERING SYSTEM

An interesting aspect of our security scheme is how layered encryption schemes are interpreted. In recent studies, researchers such as Mohammad et al.[5], and Feretti et al.[10] have shown that the use of encryption schemes can help secure an application without harming the performance for running queries or implementing the storing of data. As such, our encryption architecture aims to accomplish the same task, but by means of layering several encryption schemes together like the layers of an onion. It is important to understand that at each layer, the onion should always embody and satisfy the definition of being “an onion”. Following this metaphor, the idea of being an onion should be synonymous with the idea of maintaining a secure state. There should not be any set of combinations of encryption schemes that “unfolds” the layers of security and exposes the encryption scheme. Through each added layer, the inspected result should always be either improving the security of the ciphertext or moving laterally to maintain its security. We represent this idea using the following diagram:

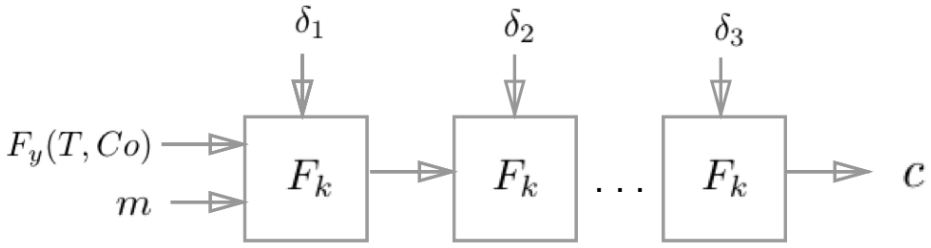
Table A: Encryption Strength Table

Strength	Classification	Repeatable?	Example
P_0	Plaintext	-	N/A
P_1	Symmetric	+	AES
P_1	Asymmetric	+	RSA
P_2	PRFs	-	Hashing

According to the table above, you first begin the encryption process with an input matching the P0 strength. At this level, there is no security, the data itself is exposed. For some client cases, this is the only necessary step for securing data. One example of this case will be table row identifiers. However, in most other cases, the client will choose how to encrypt their plaintext information. In these situations, choosing either a symmetric or asymmetric scheme is the first step in converting the plaintext information into secured ciphertext. The following step would then be to either maintain the current state of the ciphertext by encrypting with another encryption scheme of the same strength, or to finalize the encryption process by using a irreversible encryption. These sort of schemes are marked as P2 strength. By following these steps, the plaintext will always be encrypted to obtain a new ciphertext from an existing ciphertext. However, the only repeatable steps are through the use of a scheme of strength P1 . At this level, schemes such as the Advanced Encryption Standard (AES), CIpher Block Chains, Message Authentication Codes (CBC-MAC), or RSA are repeatable and do not pose a threat to exposing the original plaintext while constructing the encrypted layers. However, schemes of strength P2 have the strongest encryption strength since it is impossible to retrieve the original plaintext from it. For example, user passwords are best stored as “bcrypt hashes” in order to prevent leaking information while retaining the ability to verify a person’s password upon request by the user during operations such as account login.

ENCRYPTING DATA

The process of encrypting data is a carefully executed step-by-step procedure. Through the work of Kamara et al. in defining Cryptographic Cloud Storage techniques[6], using a clearly defined system that labels ordered procedures for the storage of information is vital to the definition of any data security system. Since these steps are labeled out in the policy file, we can determine exactly how many steps will be required. In doing so, we can represent the process of encrypting data by using a series of formulae in order to demonstrate its flexible execution.



In the figure above, we define an automaton of length n where the inputs are the message, and a function Fy that takes T and C0 as inputs. The function is used to retrieve the associated encryption keys from our Key

Store. T and C0 represents the name of the table and column from which the plaintext was derived from.

SECURING DATA

With this information, we begin encrypting based on the order specified within the policy file. We represent this using ẟn as another input to Fk . This function specifies that it either performs the encryption, or finalizes the encrypting automaton should the next scheme be a hash or irreversible scheme.

$$F_k(w, \delta, k) :$$
$$\text{if } \delta \exists \phi \rightarrow F_k(Enc_\delta(w, k), ...)$$
$$\text{else} \rightarrow Enc_\delta(w, k)$$

After n series of iterations through the automaton, we receive our final ciphertext as the output. This ciphertext can now be stored in the cloud without concern for leaking or “lossy” information.

ENCRYPTING DATA FOR MULTIPLE AUDIENCES

One of the most interesting aspects of using a multi-layered encryption scheme system is the ability to have the information serve multiple audiences at each step. Consider the following table in which each row is a layer in the resulting encryption stack:

Table B: Example of Layering Encryption Schemes

Step	Scheme	Audience
1	Plaintext	Private
2	Homorphic RSA	Accountants
3	CBC-MAC	General

At step one, we have the plaintext information. For this scenario, we will be using bank balances. The intended audience at “Step 1” is private and should only be viewed by the intended end user (i.e. User/Client). However, at “Step 2”, we transform the plaintext to ciphertext using homomorphic RSA encryption. This encryption scheme is a unique scheme in that it allows the resulting ciphertext to be used in

performing accurate mathematical calculations using the value represented in the original plaintext. As such, we provide this type of information to accountants to do their work without them ever knowing the true value of the original plaintext. Lastly, we then convert the ciphertext received from “Step 2” using CBC-MAC. This encryption scheme allows us to not only store the information securely but also adds a Message Authentication Code to verify the integrity of the plaintext when inspected. All in all, we allow three audiences to view, and operate on the same plaintext information without leaking any of the original plaintext information. This is in essence, the single most influential architectural component of the security scheme.

KEY/METADATA MANAGEMENT

For this security scheme, there are many moving components to manage. At each step of encryption, we need to know both which scheme to use and which encryption keys to use so that we can reuse it again during decryption. Keys are

used in the plural sense here since an encryption scheme could be asymmetric and require the use of two keys. In our examples, we can represent the schema for these keystore tables by using a mapping table where the index is a foreign key to a matrix table where the key is associated to specific table names and columns. We can represent these structures as follows:

Table C: Id Mapping Table

Table	Column	Identifier
Users	Salary	α_1
Users	Password	α_2
...	...	...

Table D: Key Store

Id	Keys
α_1	κ_1
α_2	κ_2

According to our tables C and D above, α is used to represent a unique identifier. We then map all α's to a κ within our key store. Each κ stored is a double-delimited string corresponding to the keys to be used (by order of encryption) for the associated column of data.

DECRYPTING MULTI-LAYERED DATA

The process of decrypting data is a carefully executed process. Since we have incorporated the process of layering encryption schemes, we never truly know exactly how many encryption schemes are being used for each column of data. However, we can know determine this information from the metadata of our parsing agent. As such, decrypting the data stored is a matter of unfurling the layers to retrieve the original plaintext. This can be best described using the replacing the executed function in the previous automaton with the following:

$$F_x(c, \delta, k):$$

$$\text{if } \delta \exists \phi \rightarrow F_x(Dec_\delta(c, k), ...)$$

$$\text{else} \rightarrow \text{return c}$$

Decrypting the data is only possible if the layers are reversible. That is to say that if one of the layers uses a hash encryption or some irreversible encryption scheme, we cannot decrypt any further.

QUERY PARSING AGENT (QPA)

After explaining the process of securing user data, we should now explain where this is all done and managed within the security scheme's architecture. These processes are primarily handled by the sophisticated "Query Parsing Agent" (QPA). An agent should be able to assist client devices with encrypting data queries and decrypting query results. Since all the data is stored and encrypted in the cloud, we need to also encrypt our queries. To explain this, we will use an example of how the agent would encrypt a simple SQL query. Let's examine the following query:

Figure E

```
SELECT * FROM Users WHERE salary > 80000;
```

In this query we are attempting to find users who make more than $80,000 annually. The job of the Query Parsing Agent here will be to focus on transforming this query to encrypt the salary information according to the policy file it was

provided. The policy would specify what encryption schemes are to be used for the salary column under the users table. After being parsed, we would end up with something similar to the following:

Figure F

```
SELECT *
FROM Users
WHERE salary > 1ae3ec7083e6;
```

Ideally, the Query Parsing Agent should be able to extract these elements to be encrypted. In this case, we have only one element, the salary number. In other cases, you may have to also encrypt table and column names. However, keywords like “SELECT”, and “FROM” should be ignored. By doing this, we can ensure that the query being passed to the cloud database to be ran will be completely secure. This will secure us against Man-In-The-Middle attacks.

THE DYNAMIC COMPONENT

One of the most exciting and powerful components to this security scheme is the ability to swap encryption keys on demand. This is handled through the layering of encryption schemes alongside the management of the keys associated to each of these layers. As such, since there is a sufficient amount of metadata management, it is possible to decrypt, and re-encrypt using new keys when required. In certain scenarios, this can be powerfully helpfully for safeguarding the remainder of existing data in the cloud after a potential threat has been resolved.

ANALYZING QUERY PERFORMANCE

When using multiple schemes, concerns are raised for query performance. However, since we follow an ordered structure for securing data, we can follow a similar ordered procedure for how to determine the performance of a query.

$$\sum_{i=1}^{n} C_{\delta_i}(m) \tag{1}$$

The formula above runs a summation through each scheme by passing the message at each step through function *C*. Function *C* is a function capable of measuring the cost, using some representable unit of time, on how long it takes to encrypt and decrypt the message. Since this measures both the encryption and decryption cost, you can take the result as the cost of going “round-trip”.

POSSIBLE ALTERNATIVE APPLICATIONS

As important as ever, securing data is vitally important to maintaining an agreeable level of trust between the client and their customers. However, time and time again, we have seen even the largest of corporate giants see security vulnerabilities. As such, we have promised this new scheme to secure data from all kinds of devices (web or mobile) in a way that allows the client to define how their data is encrypted. In one example, the client could have all their mobile apps encrypt their data using a single policy with as many layers of encryption as they wish. But another method, would be to allow users to define how they wish to encrypt their data. If done this way, in theory, each and every user would be perfectly private from each other. There are a limitless number of applications for this security scheme and it is only bounded by the ambition of the one implementing it.

FUTURE WORK

DEKSS is a powerful and unique solution for securing data within the cloud space. Currently, we are continuing our work in applying it towards various cloud platforms used by mobile devices. We also plan on analyzing the performance of our system. We hope that through these studies, we can help spread the promise of dynamic security strategy as a unique approach to combat data security threats.

ACKNOWLEDGMENTS

I would like to extend my sincerest thanks to all those involved in producing this work for providing me with the guidance, mentoring, and insight necessary. I wish to also include Iona College as none of this would be feasible without their sponsorship, and encouragement in providing a program making unique work such as this one to be produced in a creative and engaging atmosphere.

REFERENCES

[1] W. Security. Web applications security statistics report, 2016.

[2] J. K. Jung Hee Cheon. A hybrid scheme of public-key encryption and somewhat homomorphic encryption. IEEE transactions on information forensics and security: 1052-1063, 2015.

[3] M. P. S. J. Long Nguyen-Vu J.P. Privacy enhancement using selective encryption scheme in data outsourcing. International journal of distributed sensor networks: 1-7, 2016.

[4] S. M.S. B. Kamlesh Kumar Hingwe. Two layered protection for sensitive data in cloud. IEEE:1265–1272, 2014.

[5] H.S.A.Eman M.Mohammad. Enhanced data security model for cloud computing. Infos 2012, 2012.

[6] K.L.M.R. Seny Kamara. Cryptographic cloud storage. Financial cryptography and data security:136–149, 2010.

[7] G. C. Deka. A survey of cloud database systems. IEEE computer society:50–57, 2014

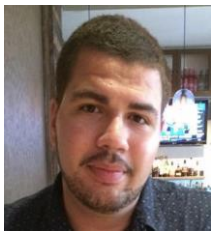
[8] Y. Y. Mahmoud Barhamgi A. K. B. Protecting privacy in the cloud: current practices, future directions. IEEE computer society: 68-72, 2016.

[9] V. S. Amitabh Saxena V. K. Application layer encryption in cloud. IEEE, APSEC:377-384, 2015. DOI: https://www.computer.org/csdl/proceedings/apsec/2015/9644/00/9644a377.pdf.

[10] M. C. Luca Ferretti F. P. and M. Marchetti. Performance and cost evaluation of an adaptive encryption architecture for cloud database. IEEE Transactions on Cloud Computing, 2, 2014

[11] T. T. Youwen Zhu Z. H. Secure and controllable K-NN query over encrypted cloud data with key confidentiality. Parallel and Distributed Computing: 50-57, 2014.

AUTHORS' BIOGRAPHY WITH PHOTOS



Stephen Rodriguez earned his undergraduate degree in Computer Science at Iona College of New Rochelle, New York, USA. He is currently completing his graduate degree in Computer Science and specializing in Software Security under the supervision of Dr. Paolina Centonze. His research has been focused on determining the security of transporting and storing data using modern web application architectures and cloud systems. This research passion stems from his extensive experience on building and working on full-stack enterprise web solutions.

Paolina Centonze has been a professor in the Computer Science Department of Iona College since August 2011. Her areas of research include language-based security and mobile computing. At Iona College, she has been responsible for extending the Computer Science curricula into the field of Cyber Security. Before joining Iona College, Dr. Centonze was a researcher at IBM's Thomas J. Watson Research Center in Yorktown Heights, N.Y. She has published extensively at numerous conferences worldwide, such as ISSTA, ECOOP, ACSAC, MDM, MOBILESoft, MobileDeLi. Dr. Centonze is also the author of two book chapters in the area of cloud and mobile security, which will appear in 2016 in books published by IGI Global and John Wiley & Sons. She is also the inventor of 10 patents issued by the United States Patent and Trademark Office.

Dr. Centonze received her Ph.D. in Mathematics and MS degree in Computer Science from New York University (NYU) Tandon School of Engineering in Brooklyn, N.Y., and her BS degree in Computer Science from St. John's University in Queens, N.Y.