Updatable Oblivious Key Management For Storage System

Sayalee Chavan

Universität Paderborn
Arbeitsgruppe Codes und Kryptographie

Contents

1 Abstract

Cryptography techniques are widely used to protect data and information being shared over the network and it also plays an important role to provide secure communication between two parties so data should not get revealed to the third party. Cryptography

uses cryptographic keys for encryption and decryption of data. Effective use of cryptography is directly proportional to a better Key Management System (KMS). KMS is a significant factor for storing user keys in big storage systems. A better KMS provides a better key life cycle management, secure communication, and also minimizes storage. While a poor KMS can lead to unrecoverable data loss. This technical paper briefly reviews how oblivious key management system mechanism provides reliability as well as efficiently stores key for the user and overcomes vulnerabilities of other mechanisms.

2 Introduction

Cryptographic keys have been effectively used for securing the business application process and are becoming more difficult to maintain. KMS processes are deployed in cloud-based operations used by Google[8], AWS [2], Microsoft [11] and more. KMS, which primarily provide keys for the security of data storage, can be categorized as symmetric and asymmetric keys. Symmetric key uses the same keys for encryption as well as for decryption, while the asymmetric keys use mathematically connected different keys for both the tasks [1]. Although larger organizations prefer to use KMS, it has some inherent security concerns and risks which should be eliminated. Cryptographic keys are needed to be generated, maintained, stored, distributed, and periodically updated [1]. Generally, KMS includes the user, the key management server, and the storage system. The user takes the help of a key management server whenever s/he wants to encrypt or decrypt the data. Storage server stores encrypted data for the user and the KM server is used for storing keys private to the user [4].

Eventually, the security of stored data is related to the transfer protocol, cryptographic mechanisms, security services provided by cryptographic mechanisms, and robustness of the key [1]. Transfer protocol is used for the security of information being transmitted from the client to the server. In the case of key management, the security of cryptographic keys is important while transmitting from the user to the key management server. Usually, the Transport Layer Security (TLS) protocol is used for secure communication but attacks on TLS can lead to exposure of cryptographic keys [4]. Some examples of cryptographic mechanisms are symmetric encryption scheme, asymmetric encryption scheme, and hash functions. These mechanisms provide security services mainly used for the protection of keys[1]. Some security services like authentication referring to providing identity, confidentiality meaning information should not be revealed to the third party, authorization meaning only valid parties will get access to sources, information and keys, and lastly, non-repudiation meaning the validity of something cannot be denied. Besides it relates to the digital signature and digital certificates which can be used for identity and validation [1]. Furthermore, a key strength is the main factor to be considered for key security [1]. However, the basic KMS does not ensure strong security against attacks, which creates the need for a stronger KMS.

Oblivious key management system (OKMS) offers more security features than a basic KMS, and also overcomes vulnerability existing in basic KMS. It provides the Oblivious Pseudorandom Function (OPRF) feature which is used to hide the key from the KM

server [4]. The next extended version of OKMS is the Updatable Oblivious KMS which provides update functionality for keys, so it can regularly update keys whenever needed [4]. Therefore, it is of interest to see the process and implementation behind the Basic KMS, Oblivious KMS, and Updatable Oblivious KMS in more detail.

3 Wrap-Unwrap process

A key wrap-unwrap algorithm is needed for protecting keys from unsecured storage and transfer. It, being a trivial method, is utilized by many business processes. Whenever the user needs to encrypt or decrypt the data object, he uses the data encryption key (DEK) [4]. However, he utilizes the service of the Key Management server to keep DEK secured. Wrapping process wraps DEK at the server side which is used for encryption of data object. For the protection of DEK, wrapping is required. Similarly, DEK needs to be unwrapped for decryption of data object[4].

3.1 Wrapping process

Figure 1 describes wrapping process. This process includes three steps as follows:

- Firstly, the user encrypts the data object by making use of randomly chosen symmetric DEK and sends this DEK and data object ID to the KM server. Before wrapping DEK, the KM server authenticates the user first and then wraps the DEK using user-specific key K(u) [4].

- The KM server gives a response back to the user with data object ID and Wrap(DEK)[4].

- Once the user gets wrap from the server, immediately user transfer Wrap(DEK, K(u)), data object ID, and encrypted data object with DEK to the storage server for storage[4].

3.2 Unwrapping process

Figure 2 describes unwrapping process. This process includes the following steps:

- When a user wishes to decrypt the data object, he fetches data Wrap(DEK, K(u)), data object ID and encrypted data object from the storage server[4].

- The user makes a unwrap request to the KM server by sending the attached wrap and data object ID. After receiving a request, the KM server authenticates the user and sends back data object ID and unwrapped DEK to the user [4].

- Once the user receives unwrap DEK from KM server, he can decrypt data object with DEK he retrieved from the server [4].

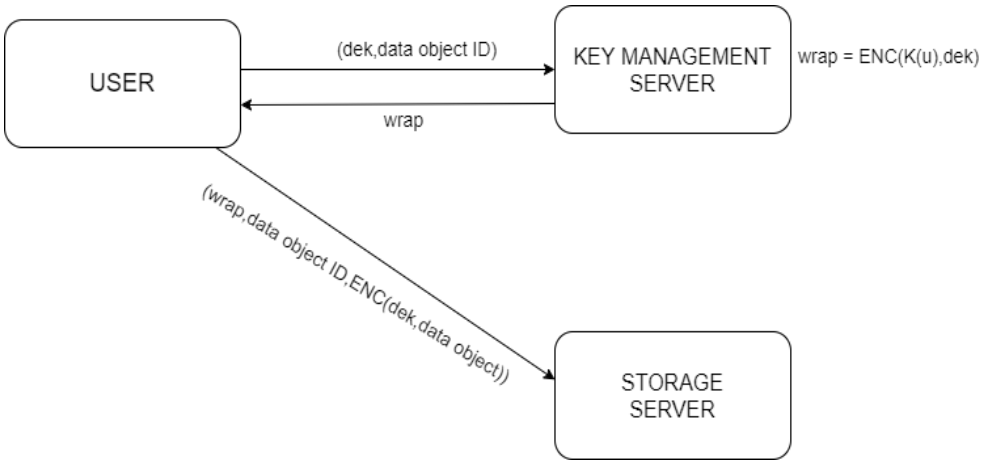

Figure 1: Wrapping process [4]

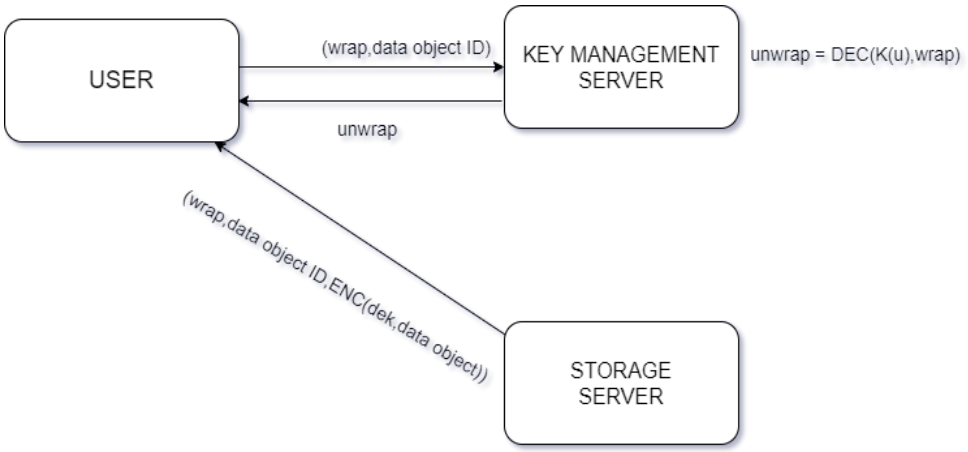

Figure 2: Unwrapping process [4]

3.3 Vulnerabilities

Most of the communications are vulnerable to attacks and attacks are possible while transmitting DEK from a user to the KM server or vice versa [4]. Some security issues related to KMS are as following:

- Security of DEK is related to TLS which is used for protection of transmission channels between the user and server[4].
- TLS can easily get attacked, because of configuration or implementation errors to certification. The hackers can also tear the protection and try to get access to DEK but sometimes TLS gets terminated prematurely which can lead to exposure of DEK to attackers [4].
- Reliability of data is the main concern in KMS if an application crashed maliciously or accidentally which can cause irrecoverable data loss[4].
- The server can see DEK received from the user as server-side wrapping requires communication between the server and DEK[4].

4 Oblivious Key Management System

According to [4] the OKMS is based on OPRF. OKMS provides a better KMS which overcomes vulnerabilities of the wrap-unwrap approach [4]. A pseudorandom function (PRF) is an efficiently computable keyed function whose values are calculated from random elements in the function range and are indistinguishable for a randomly selected key k [5]. Suppose, K is a key, and f(K,x) is a function for K and input x. Consider g is a truly random function. Function f is the pseudorandom function if f is computable in polynomial time and output of f(K,x) and output of g(x) is indistinguishable where both the functions apply on the same input[6]. OPRF protocol follows oblivious transfer where the sender is guaranteed the receiver is not receiving more details than allowed and the receiver is ensured the sender does not know which part of the inputs it is receiving [10].

4.1 Oblivious Pseudorandom Function

OPRF is a PRF between two parties where the one party is user and the other party is oblivious key management (OKM) server[4]. However, the input and output are visible to the user but not the key. On the other hand, OKMS can see the key but is restricted from seeing the input and output. In short, OPRF hides the key from the user as well as input and output from the OKM server [4]. Figure 3 describes basic functionality of oblivious key management systems. This process has following steps:

- The user first encrypts x with some blinded factor B (for invisibility of input x), then user sends B(x) to OKMS server[4].

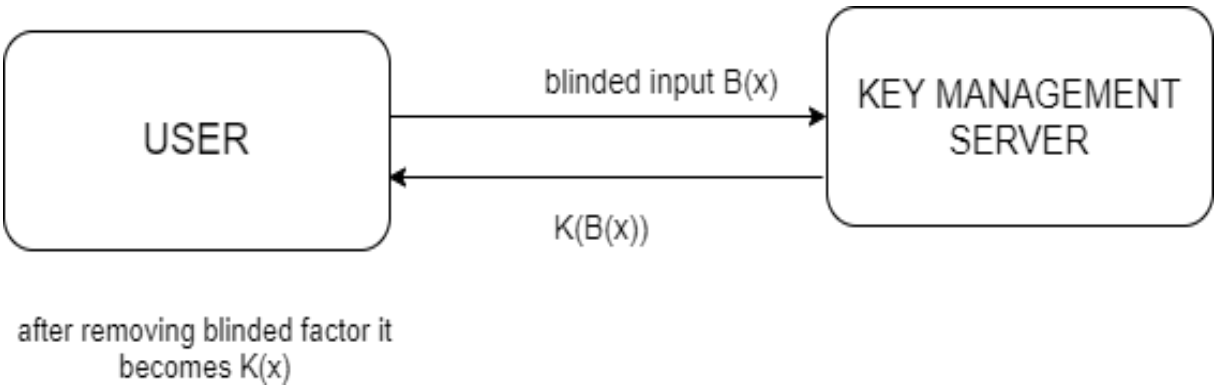

Figure 3: Basic functionality of Oblivious key management [4]

- Once the input is received OKMS, it encrypts input with key K so input becomes K(B(x)) and send it back to the user. Due to blinded factor, input x is not visible to OKMS server[4].

- After receiving a response from OKMS server, the user removes blinding factor B from x and it becomes unblinded version K(x). At the end of communication user receives input encoded with OKMS key[4].

4.2 OKMS implementation

Oblivious KMS is a better alternative for a wrap-unwrap approach. Unlike the wrap-unwrap scheme, OKMS does not depend on the security of TLS, DEK protection is provided by features of the OPRF function[4]. In the OKMS scheme, the key management server stores user-specific OPRF keys K(u) where the user encrypts data by using a symmetric encryption scheme and DEK but the user takes the service of OKM server to generate DEK and sends blinded data object ID to KM server. Then OPRF function computes DEK with data object ID and OPRF key provided by the server [4]. Hence, the user sends a blinded data object ID that is not visible to the server and key K(u) is invisible for the user [4]. Subsequently, DEK is generated using OPRF computation with data object ID and K(u) as input. Once DEK gets generated, the user encrypts the data object under DEK by using a symmetric authenticated encryption algorithm and stores data object ID and encryption at the storage server [4]. In the case of decryption, the user retrieves data object ID and encrypted data from the storage server. DEK can be retrieved with the same OPRF computation done in the encryption scheme and data object can be retrieved [4].

Thereupon, OKMS eliminates major vulnerabilities present in the basic KM system and strengthens the security properties. Firstly, DEK revelation to sever, since the OPRF function does not allow DEK visibility to the server. Secondly, the security of keys while transmitting from user to server[4]. In OKMS, the key transmission protected by the OPRF feature, so security does not depend on TLS[4]. If we are concerned about

attacks, though attacker try to hack data in transmission, he will only learn about blinded data object ID as it does not reveal any important information[4]. OPRF also includes key verifiability property that the user can verify returned DEK is valid or not, as it will lead to irrecoverable data loss if DEK is wrong[4]. Additionally, Diffie-Hellman (DH) based OPRF provide advanced properties than normal OPRF scheme. In the DH protocol, the sender and receiver allowed to generate a mutual secret key over insecure communication, where an attacker or third party is not able to compute secret key easily[9]. DH-based OPRF makes sure that unauthorized parties will not able to see data encryption keys and data object IDs[4].

5 Updatable Oblivious Key Management System

Updatable Oblivious KMS(UOKMS) is an extended version of Oblivious KMS where "Updatable" refers to update user-specific master key K(u) stored at KM server[4]. In cryptography, some keys are using for a longer time to protect the data, which is common in large storage systems but the main property of this scheme is to reveal very little data if key get exposed[4]. Changing key periodically will reduce the risk of visibility of the master key to the attacker. This procedure is also referred to as Key Rotation in cryptography[4]. UOKMS adapts updatable encryption scheme to upgrade user-specific master key. Though the attacker can learn K(u), he can only see encrypted data but he will not be able to decrypt it as ciphertext has already been updated with new key K'(u)[4]. Using the same key rotation technique in the basic wrap-unwrap approach will cost time as it will include unwrapping and then re-wrapping of all the ciphertexts at storage server with newly generated user-specific key[4]. In comparison, UOKMS is a more enhanced scheme that provides better security and performance than the basic wrapping approach[4].

5.1 Atomic Key Rotation

In UOKMS, key rotation is an important feature as it updates the user-specific master key[4]. Generally, in case of a wrap-unwrap process, the user has to send each wrap to the server and then the server will generate new master key for the user, and then all the DEK should get wrapped by new user-specific key and send it back to the user, he will again store this updated wraps in the storage server. This process is quite time-consuming and also has some performance issues[4]. On the other hand, in UOKMS, this technique works much better than wrap-unwrap.[4] The process involves the following steps:

- The user makes rotation requests to the KM server[4].
- Once request is received by the server, it generates new key K'(u) and computes delta (D) which is nothing but K'(u)/ K(u). After computing D, the server deletes old key K(u)[4].

- The server forwards D to storage-server, in the background storage server starts updating all wraps with delta present in storage server[4].

It is observed that we do not need interaction between the user and KM server as the user only makes key rotation requests, moreover, the key transmission is not required which reduces the risk of exposure of keys[4]. In this model, communication establishes between KM sever and storage server for exchanging delta only. Additionally, the storage server updates all wraps locally and delete D value at the end of update[4].

5.2 Updatable Encryption

Updatable encryption(UE) scheme [7] is the best scheme to use for key rotation as it updates ciphertext from the old key to new key provides stronger security model. Updatable encryption schemes allow full key rotation without decryption. Ciphertexts created under one key can be rotated to ciphertexts created under another key[7]. It guarantees protection of user-specific master key where the single token which is a division of new key and old key is used for updating ciphertext. In cryptography, it is considered a good procedure for use[7]. UOKMS makes use of updatable encryption properties also adapts it's security features and besides, the use of token value for updating key. The delta should not carry sensitive information that can be harmful to a user[4]. Furthermore, if we consider security features, it also comes up with post-compromise security and forward secrecy where in terms of confidentiality, ciphertexts updated by new key should remain indifferentiable as attackers should not differentiate between ciphertexts or get information on the age of ciphertexts.[4]

UE scheme [7] provides post-compromise security, if key revealed to the attacker but ciphertext is already updated with a new key, the attacker will not able to decrypt data as ciphertext has updated with a fresh key[4]. Forward security means if attacker learned about old ciphertext and has a new key to decrypt data, an attacker can't learn anything from ciphertext as it is old ciphertext which is already updated in the storage server[4].

5.3 UOKMS implementation

As UOKMS is an adaptive version of the Oblivious Key Management system, so it extends obliviousness for security purposes which ensures the safety of data encryption key by OPRF function[4]. As we learned from the OKMS scheme, KM servers always store user-specific key k(u) and Key rotation allows the server to produce a new key which can be used to update all wraps present in the storage server[4]. Suppose, Y(u) = $g^{K(u)}$ where g is a fixed generator stored at storage server and this scheme has following steps [4]:

- Firstly, the user encrypts data object under public key Y(u), which corresponds to user-specific key K(u) stored at updatable KM server. Ciphertext includes data object ID, wrap, and encryption of data object under DEK. At the time of encryption, the user sets f = g^l where l is a random element and then computes

DEK from $Y(u)^l$. This does not need communication with the KM server as encryption done locally by the user. At the end, Ciphertext becomes ci = (data object ID, f, enc(data object, DEK)) [4].

- For decryption of ciphertext, we need oblivious interaction with KM server where the user computes $f^{K(u)}$ and derives DEK from it. The correction of DEK will be done with $Y(u)^l = f^{K(u)}$. So this can be proved as $Y(u)^l = (g^{K(u)})^l = (g^l)^{K(u)} = f^{K(u)}$ [4].

- In the end, When key rotation and update are requested from user, process explained in section 5.1 will take place, and the update of ciphertext will be at done at the storage server without any interaction [4].

It is observed that the user only needs the KM server for decryption of data object as opposed to the OKMS scheme which reduces risks[4]. For encryption, the user derives DEK from public key value, and for decryption, the user derives DEK interaction with the user obliviously because this correction will take place by verifying both the DEK generated at encryption and decryption time[4]. Though UOKMS uses public-key encryption because decryption of data objects is only possible by the authorized users only.[4] This scheme has various algorithms, key generation algorithms used by the KM server to produce the user-specific key, symmetric authenticated encryption algorithm for encryption and decryption of a data object, update generation algorithm for generating a new user-specific key and ciphertext update algorithm for updating ciphertext in the storage server.[4]

Another advantage of UOKMS, it enhances performance as an update of ciphertext is done locally in storage sever.[4] According to the UE scheme[7], computation for the update process requires one exponentiation as it does not depend on the size of the ciphertext. Therefore, this process does not take much time to complete the update. Since encryption is non-interactive, no need to verify the key. In the case of decryption, key returned by KM server gets verified by symmetric authentication decryption scheme.[4]

5.4 Security prototype of UOKMS

Generally, a security prototype should provide strong features for protection purposes of the key[4]. Initially, data can be encrypted under user-specific key K(u) but after the key rotation process, encryption and decryption must be under K'(u)[4]. if we consider this process as periodically then, for first times-pan t1, ciphertext c1 updated by k1, in second time-span t2, ciphertext c2 updated by k2 and so on. Accordingly, each time-span has different keys that protect ciphertexts and every next time-span has an updated version of the previous ciphertext[4]. This model provides security against post compromises and future attacks[4]. Two cases can occur:

- Suppose, Attacker has key k1 generated in time t1 and he got access to ciphertext c2, still it should not be possible for an attacker to decrypt c2 because c2 can only decrypt by key k2[4].

- If an attacker has the ciphertext c1 and key k2, it would not be possible for an attacker to decrypt it where c1 is already updated in the storage server by new key k2[4].

This way, the attacker must have information about updated key to decrypt the ciphertext. However, It is difficult for an attacker to keep this information updated and get recent updated key [4]. Therefore, the attacker should have an updated ciphertext with an updated key. Conversely, the third case might occur if the attacker succeeds to gain the right key for decryption of ciphertext then he will get the access of data[4].

6 Comparative analysis of KMS, OKMS, UOKMS

KMS	OKMS	UOKMS
Key Management System is a simple traditional approach used for storing cryptographic keys(wrap-unwrap scheme) and does not provide the security of keys in transit.	Oblivious Key Management System uses the OPRF function to ensure that, the key is not visible to the server.	Updatable Oblivious Key Management System adapts the Upadatable Encryption scheme and updates all the ciphertexts with newly generated keys.
Lacks the security of DEK.	Better alternative for KMS.	Adaptive version of OKMS provides enhanced security for DEK.

7 Conclusion

This report highlights different KMS in cryptography. The KMS is simple to use and implement but does not give a security guarantee for keys since the key is an important factor for the user. As we learn from the wrap-unwrap approach, the key is visible to the server and it can cause exposure of key to the attacker. Transmission channels used in this approach are easily vulnerable to attacks which can lead to the revelation of the key. These two weaknesses are taken care of in OKMS which uses the OPRF function to generate key for users to encrypt and decrypt data. The main feature of the OPRF function is that it does not reveal the key to the server which provides enhanced security, as no one can learn about the key. Finally, the updatable oblivious key management system is an extension of the oblivious key management system, which makes use of the updatable encryption technique for key rotation and adapts obliviousness for communication. In this scheme, the server produces a new key to regularly update the ciphertext which contains encrypted data and the key which is used for encryption of data. The security features of these schemes will make a good model for the key management system.

References

[1] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management part 1: General (revision 3). *NIST special publication*, 800(57):1–147, 2012.

[2] Matthew Campagna. *Aws key management service cryptographic details*, 2015. https://d0.awsstatic.com/whitepapers/KMS-Cryptographic-Details.pdf.

[3] Adam Everspaugh, Kenneth Paterson, Thomas Ristenpart, and Sam Scott. Key rotation for authenticated encryption. Cryptology ePrint Archive, Report 2017/527, 2017. https://eprint.iacr.org/2017/527.

[4] Stanislaw Jarecki, Hugo Krawczyk, and Jason Resch. Updatable oblivious key management for storage systems. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 379–393, 2019.

[5] Stanisław Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography Conference*, pages 577–594. Springer, 2009.

[6] Burt Kaliski. *Pseudorandom Function*, pages 485–485. Springer US, Boston, MA, 2005.

[7] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 685–716. Springer, 2018.

[8] John Houston Lowry and Jonathan A Rubin. Cloud key management, January 16 2014. US Patent App. 13/545,805.

[9] Ueli M Maurer and Stefan Wolf. The diffie–hellman protocol. *Designs, Codes and Cryptography*, 19(2-3):147–171, 2000.

[10] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[11] Rami Vemula. Handling secrets and exceptions in azure functions. In *Integrating Serverless Architecture*, pages 203–237. Springer, 2019.