10

Comparative Analysis of Master-Key and Interpretative Key Management (IKM) Frameworks

Saman Shojae Chaeikar, Azizah Bt Abdul Manaf and Mazdak Zamani
Universiti Teknologi Malaysia
Malaysia

1. Introduction

The process of generating, distributing and revoking a cryptographic key is called key management (Piper & Murphy, 2002; Fumy & Landrock, 1993). Key management is a very challenging area between cryptographers and attackers, and since finding keys of approved cryptographic techniques computationally is impossible, attackers prefer to somehow breach the key management process instead of trying to crack keys (Techateerawat & Jennings, 2007).

Today many techniques like DES, 3DES, RSA etc. have emerged for protecting data secrecy. One internationally approved and widely used key management method is master-key which empowers a computer to generate keys from a Key Derivation Key (KDK), the parties' identity, labels and some other information (Chen, 2009). By utilizing master-key a computer can generate new keys and distribute them among its nodes, sometimes even establishing a hierarchy among them (Onmez, 2009).

IKM has developed from performing some changes on workflow and key generation algorithms of master-key to achieving more dynamism and some advanced features like intelligent attack resiliency (Chaeikar et al., 2010b, 2010a).

Master-key and IKM have many features in common. In both there are algorithms of key generation, both use environmental parameters as key generation parameters, both generate symmetric keys, both generate keys of variable length and so on. Therefore it was felt that there was a need for guidelines on both schemes to facilitate making a choice regarding requirements. The purpose of this article is to provide a guideline for helping researchers and developers to choose the proper cryptographic scheme between master-key and IKM regarding their goal of application and working criteria. To do so, master-key and IKM are compared according their security features and amount of imposed traffic load on the server and network.

The second section of this chapter explains the process of master-key key generation in three modes. The third part discusses IKM key generation workflow and algorithms concisely. The fourth part compares these two methods in terms of security and performance to help researchers and developers to choose the most convenient technique between master-key and IKM with regard to their required features and criteria. The last section summarizes the results of this analysis in a few paragraphs for fast review of this article.

2. Master-key

A key derivation function is a function that uses an input key and some other input data to produce keying material which will be employed by a cryptographic algorithm. A key that is input of key derivation function is called a Key Derivation Key (KDK). This key either can be generated by an automated key generation process (Piper & Murphy, 2002; Fumy & Landrock, 1993) or by an approved random bit generator function (Onmez, 2009). If a KDK is generated through an automated key generation process then it will be considered as part of secret keying material of that process.

Any chosen part of derived keying material which meets the needed key length can be used as input of cryptographic algorithm. To ensure that all parties using the same Key Derivation Function (KDF) are synchronized they must agree on the way that the keying material will convert into a cryptographic key. For instance, if derived keying material length is 256 bits then the first segment (first 128 bits) can be used as an authentication key and the second segment (second 128 bits) as an encryption key.

If KDF uses Pseudo Random Function (PRF) then based on desired length of keying material the KDF might require calling PRF several times to achieve the desired length. This article explains three types of master-key key generation modes. The following notations are described concisely for a better understanding of algorithms.

K_I: a key which will be fed into a KDF for deriving the keying material.
K_O: binary string output of KDF.
Label: a binary string that explains the goal of key derivation.
Context: a binary string including information such as identities of parties which are deriving/using derived keying material and sometimes a nonce which is known by those parties who derived the keys.
IV: initial binary string value of first iteration of feedback mode which either can be kept public or secret. IV also might be an empty string.
L: an integer which shows length of derived keying material K_0 in bits. Length of binary string varies based on the encoding method of input data.
h: an integer which shows length of PRF output in bits.
n: an integer which shows the number of needed PRF iterations to achieve L bits of keying materials.
i: binary string input of PRF in each iteration.
r: an integer smaller or equal to 32 which shows binary length of counter i.
{X}: shows that the value of X is an optional value for KDF.
0x00: separator of different parts of variable length data fields.

A PRF type KDF concatenates n times output of PRF until it achieves the L bits of desired length where $n=\lceil L/h \rceil$. In counter mode n should not exceed $2^r - 1$ while $r \leq 32$. For double-pipeline iteration and feedback mode maximum value of n is $2^{32} - 1$. In every iteration of PRF the KDK K_I will be used as key and the input data includes both fixed input data and an iteration variable. Iteration value depending on iteration mode can be a counter, the result of last iteration of PRF, a mix of both or the result of first iteration if the mode is double pipeline.

2.1 Counter mode key derivation function (KDF)

Counter mode result of PRF will use a counter as iteration value. Counter mode is structured as follows:

h: PRF result length in bits.
r: length of binary format of counter i.

Input data: K_I, L, Context, and Label *Output*: K_O

Counter mode KDF algorithm:

1. n=⌈L/h⌉
2. if $n > 2^r - 1$ then error
3. result(0):= ∅
4. for i=1 to n do
   4.1. K(i):= PRF(K_I,$[i]_2$||Label||0x00||Context||$[L]_2$)
   4.2. Result(i):= result(i-1)||K(i)
5. Return: K_O:= first L bits of result(n)

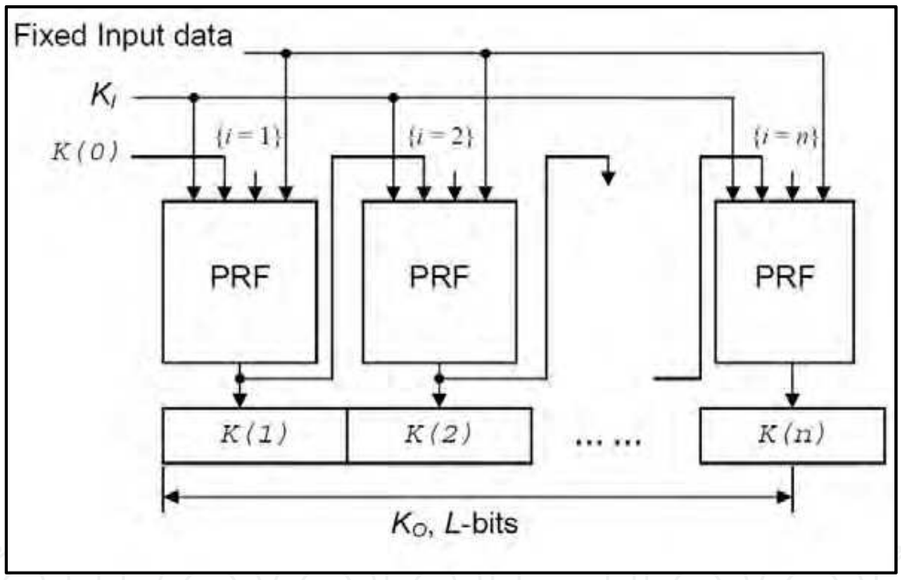


Fig. 1. Counter mode KDF structure (Chen, 2009)

In every round of iteration constant input values are Label||0x00||Context||$[L]_2$ and iteration variables are K(i-1){||$[i]_2$}. The structure of counter mode is illustrated in Fig. 1.

2.2 Feedback mode key derivation function (KDF)

In feedback mode the result of the PRF is generated according the results of last iteration and counter of number of iterations. Feedback mode is structured as follows:

h: PRF result length in bits.
r: length of binary format of counter i.

Input data: K_I, L, IV, Context, and Label. *Output*: K_O.

Feedback mode KDF algorithm:

1. n=⌈L/h⌉
2. If n > $2^{32} - 1$ then error
3. Result(0):= ∅ and K(0):= IV
4. For i = 1 to n, do
   4.1 K(i):= PRF (K_I,K(i-1){||[i]$_2$}||Label||0x00||Context||(L)$_2$)
   4.2 Result(i):=result(i-1)||K(i)
5. Return: K_O:= first L bits of result(n)

In every round of iteration constant input values are Label||*0x00*||Context||$[L]_2$ and iteration variables are *K(i-1)*{||$[i]_2$}. The structure of feedback mode is illustrated in Fig. 2.

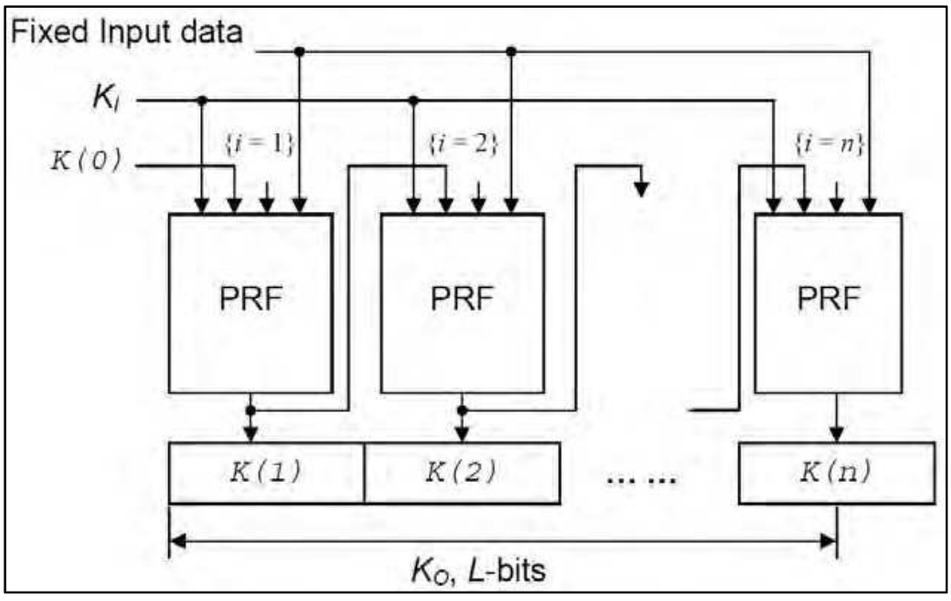


Fig. 2. Feedback mode KDF structure (Chen, 2009)

2.3 Double pipeline iteration mode key derivation function (KDF)

In counter mode or feedback mode the PRF iterates in a single pipeline while in double pipeline iteration mode there are two pipelines. In the first pipeline series of secret values A(i) will be generated which then will be fed into respective PRF iteration of the second pipeline. Double pipeline iteration mode is structured as follows:

h: PRF result length in bits.
r: length of binary format of counter i.

Input data: K_I, L, Context, and Label *output*: K_O.

Double pipeline mode KDF algorithm:

1. n= ⌈L/h⌉
2. if n ≤2^{32} then error
3. result(0):= ∅

4. A(0):=IV=Label||0x00||Context||$[L]_2$
5. For i=1 to n do

5.1 A(i) := PRF (K_I, A(i-1))

5.2 K(i) := PRF (KI, A(i){|| $[i]_2$}||Label||0x00||Context||$[L]_2$)

5.3 Result (i) := result(i-1)||K(i)

6. Return:K_0, i.e., first L bits of result(n)

The first iteration pipeline uses the result of feedback mode with the initial value of A(0)=IV=Label||0x00||Context||$(L)_2$. K(i) by using A(i), and as an optional choice, counter $(i)_2$ are iteration variables in every iteration of the second pipeline. The following figure illustrates double pipeline KDF.

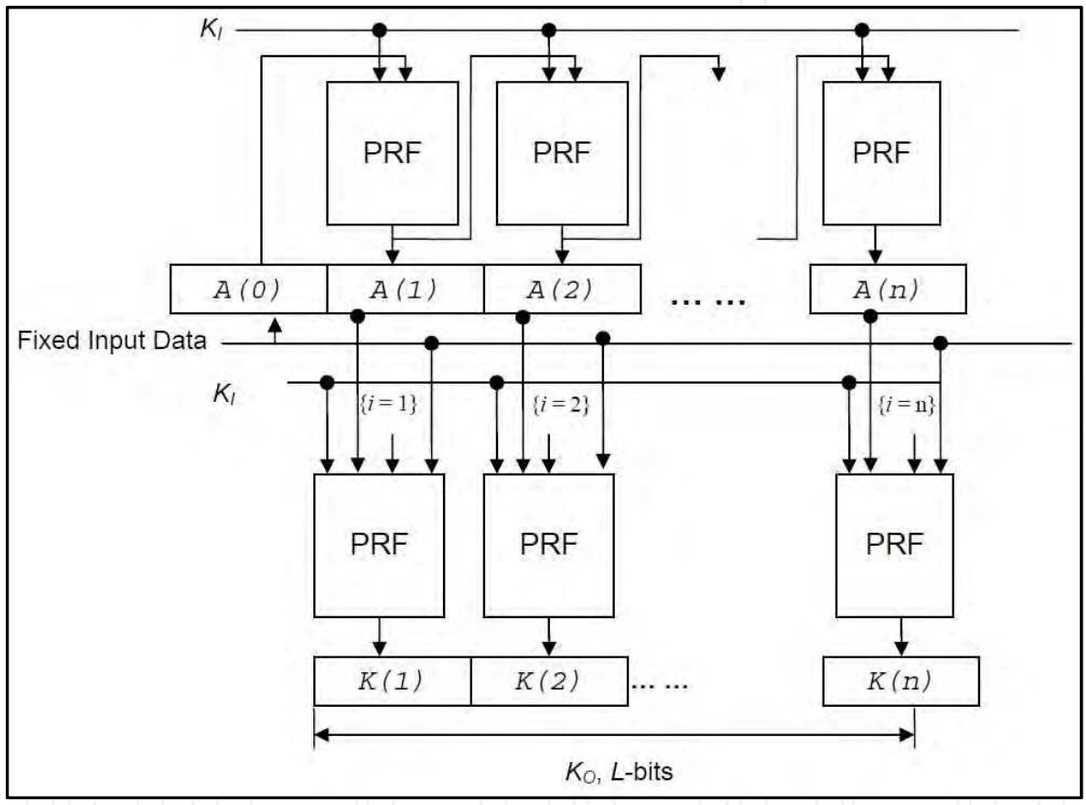


Fig. 3. Double pipeline mode KDF structure (Chen, 2009)

2.4 Key derivation key (KDK) length

In some of KDFs length of KDK depends on PRF. For instance, if Cipher-based Message Authentication Code (CMAC) was being used as PRF then key length directly depends on length of the respective block cipher. Therefore, at application time, consistency of PRF and KDK must be checked.

In contrast to CMAC if Keyed-hash Message Authentication Code (HMAC) was being used as PRF then the KDK can be almost any length. To provide consistency of PRF output and block length, if the key is longer than hash function block length then the key will be hashed into the length of the hash function output.

2.5 Conversion of keying material into cryptographic keys

The derived keying material length depends on the chosen cryptographic algorithm. The algorithm that will apply generated cryptographic key, like message authentication code, will determine its length. If no limitation is defined by the application then any portion of derived keying material, which has needed length, can be employed as a cryptographic key only if there is no overlapping among derived keys from KDF output. Therefore, length of derived keying material must be more or equal with the sum of keys.

3. Interpretative key management (IKM)

Core of Interpretative Key Management (IKM) workflow is a key server which is responsible for:

Producing the interpreter
Distributing the produced interpreter among nodes
Supervising nodes' activities for higher security
Declaration of revocation of the interpreter to nodes when it expires

The first responsibility of the server is generating the interpreter to be distributed among the nodes. The interpreter includes a time zone, a calendar, a bit-stream source address, a 24 digit number, a key generation algorithm and a revocation code. Once the interpreter has been created then it must be distributed among authorized nodes through a secure channel. When the interpreter is installed on nodes then the first 512 bits of the defined file from the given address will be downloaded. Since keys are time-dependent and key generation components which are bit-stream source, defined time zone and calendar, 24 digits and key generation algorithm are the same among all computers and they will be able to generate identical time-dependent keys in future without any coordination or key distribution. Also because keys have a predefined lifetime then without a key revocation call they will expire.

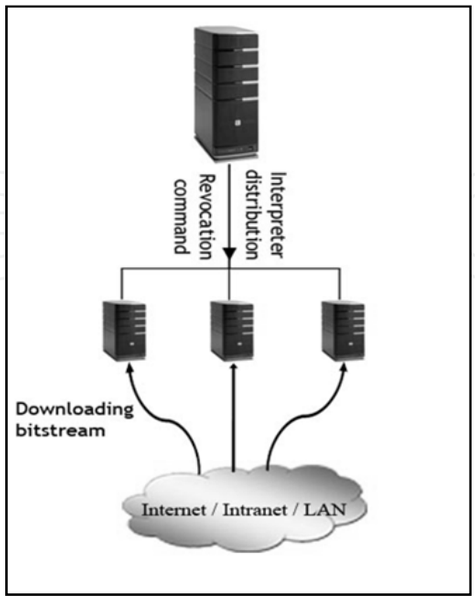


Fig. 4. Interpretative key management (IKM) workflow

One of the main key management issues is the existence of key storage, but in IKM because keys will be generated when they are needed and keys are constantly changing then no key storage is required. IKM keys expire automatically after the lifetime expires and a new one will replace automatically. But once the server issues a revocation call this means that the interpreter is expired and generated keys will no longer be valid. The following sections explain the detailed responsibilities of IKM components.

3.1 Server

The server role starts with generating the interpreter. To generate it a time zone, a calendar, a bit-stream source, a revocation code, a key generation algorithm and 24 digits must be selected and embedded into it. Once it has been generated then through an established secure channel, like Diffie-Hellman, it must be distributed among authorized parties.

Every generated key will be expired when its lifetime expires, but the interpreter will continue working until it receives a revocation call which means one of the interpreters is either compromised or is suspected to be compromised. In such a case all nodes will stop using the current interpreter and will wait until they receive a new version.

Once the interpreter has been generated and distributed then the server will supervise the security status of nodes. Because the interpreter is in charge of encryption and decryption, and keys are time-dependent then it can distinguish genuine packets from attack packets and report the number and type of attacks to the server. The server, based on received security reports, will decide whether the node can continue working, temporarily must stop working or must shut down completely.

3.2 Interpreter

The interpreter's task is to generate fresh synchronized keys and to send security status reports to the server. The interpreter's components and important points are as follows.

3.2.1 Bit-stream source

The bit-stream source is the address of a file on the network, the first 512 bits of which will be downloaded to construct a bit matrix to be fed into the key generation algorithm. The first 512 bits of the file, regardless of file type, will be downloaded and arranged into a 22*23 matrix. One of the most important criteria for choosing a bit-stream source is its updating intervals which, if not long enough, will result in inconsistency of keys among different interpreters because they may download different bit-streams at different times and eventually generate different keys. If the IKM being deployed is on a WAN then a shared file via FTP service can be utilized as a bit-stream source.

The bit-stream source can either be placed on the network or can be a file with its update intervals known by the administrator of the established encrypted sessions. For instance, if the administrator decides to use the available sources then an online PDF version of a newspaper which updates at particular times would be the proper choice. In this case the chosen source will be changed every day at a particular time and all nodes must update their bit-stream every time it changes. But if the updating interval of the chosen source is longer than our needed time or the source is located on the server, then renewing the bit-stream is not necessary.

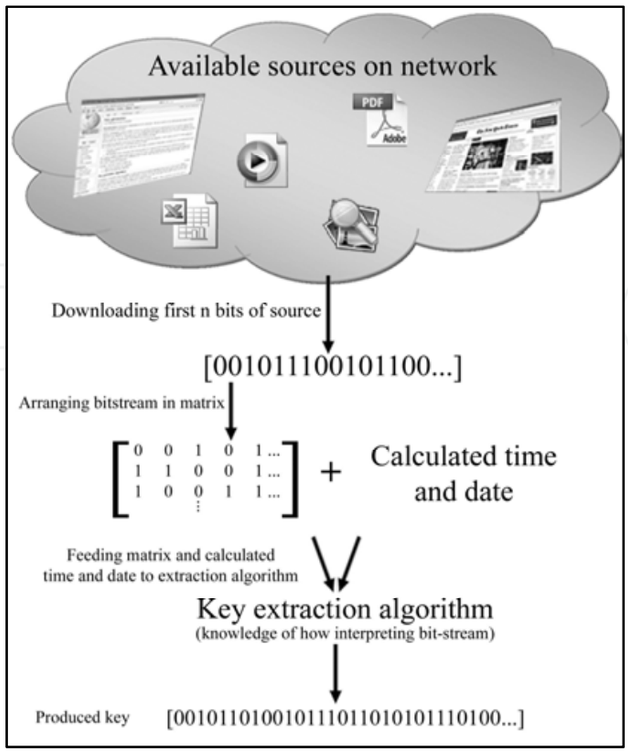


Fig. 5. IKM key generation process

3.2.2 Time, date and 24 digits

Date and time are two parameters used in the process of synchronized key generation, but because nodes might be distributed around the world then they will use different regions' time and calendar. So to unify the time and date of all nodes they must agree to the same time zone and calendar. For instance, the UTC time and Gregorian calendar can be chosen so that computers have the same parameters. Then each computer will compute the difference of time and date to be used at the key generation process.

Time and date unification is a very important process because firstly using time and date will lead to having time-dependent keys that can resist against attacks like replay attack. Secondly if there are any problems for a node in synchronizing itself then it will not be able to communicate with other nodes. The IKM server, designated web site or any publicly available time server can be utilized for synchronizing nodes.

From the viewpoint of attackers, guessing date and time is a very easy step to achieving two important key generation factors. Therefore, for tightening IKM key generation security, full format time and date will be repeated twice to achieve 24 digits. Then the 24 embedded digits will be added to the time and date 24 digits to make it unbreakable for attackers. For example, if the current date and time is 27/7/2010, 5:44 then its twice repeated full format will be 270720100544270720100544. Then the 24 randomly generated and embedded digits will produce 24 new digits that are not guessable by attackers and will change every minute. Fig. 6 illustrates process of converting date, time and 24 digits into new 24 digits.

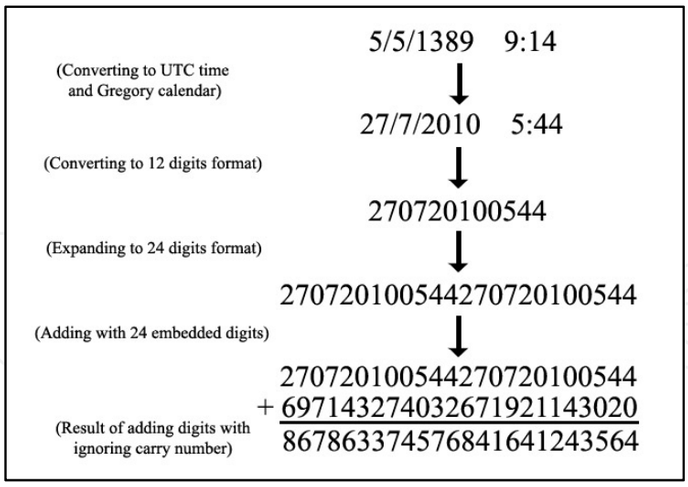


Fig. 6. Conversion process of current time and date into 24 digits

3.2.3 Revocation code

The revocation code is an embedded secret code in the interpreter that once received from the server means that the current version of the interpreter either is compromised or, according to the decision of the IKM server administrator for security reasons, is no longer valid and must stop working. All nodes will send their security reports to the IKM server for monitoring purposes and regarding received reports it can decide whether the current version is compromised, is suspected of being compromised or can continue working. Also for enhancing security it is better to renew interpreter periodically.

3.2.4 IKM key lifetime

Regarding the IKM structure and level of desired secrecy, three types of key lifetime are introduced in IKM. Because of utilizing time and date in key generation process, period of updating keys can be every minute, hour, or day.

A new key will be produced and employed every minute, hour or day. If a new key every minute is chosen then a new key will be generated and employed every minute. If the hourly key is selected then a specific minute of every hour will be considered as the parameter of key generation regardless of whether the nodes are going to join before or after it. For a daily key, a specific hour and minute will be considered as the time parameter of the key generation process.

Early seconds of changing key, both new and previous keys will be valid for decryption. Those packets which are encrypted and sent at the last moments of the changing minute will most likely be received in the early seconds of the following minute. Therefore, those packets are encrypted with the last key and cannot be decrypted with the new key. The maximum time needed for sending a packet between two distant nodes will be the time of double key checking. Once a packet is received in the first seconds of the changing key firstly an attempt will be made at being decrypted with the new key, but if this process fails then it will be decrypted with the last key. If neither new key nor the last key could decrypt it then this packet runs the likelihood of being a running replay attack against the node and

if it is repeated many times then it will be reported as an attack to the IKM server. Fig. 7 shows the process of decryption of received packets.

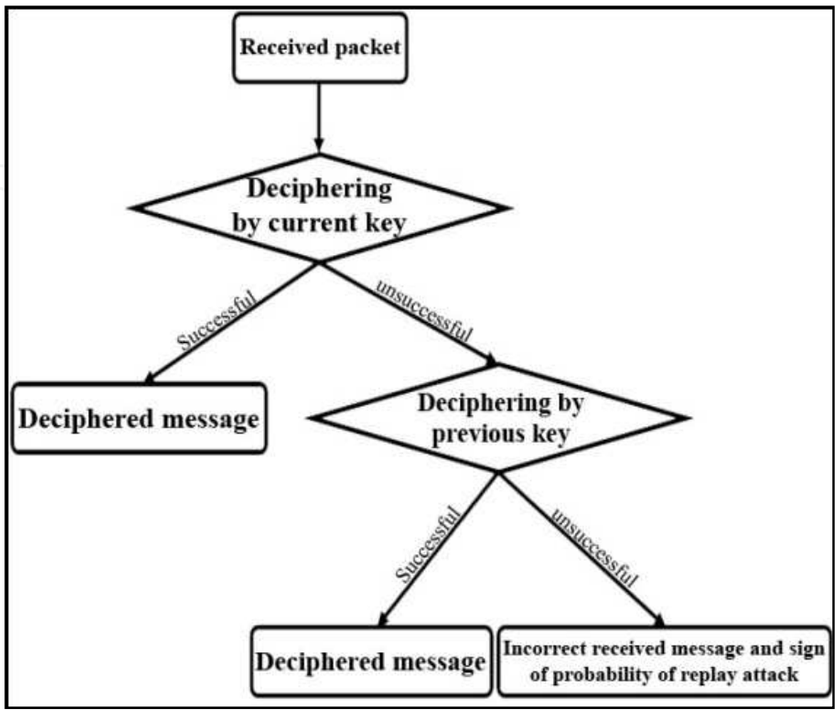


Fig. 7. Double key checking diagram

3.2.5 Key generation algorithm

IKM utilizes many key generation schemes and one of them is the matrix technique. The matrix technique is downloading a bit-stream, arranging it in a matrix and surveying the matrix according to 24 generated digits. The 24 digits explain how the matrix must be surveyed from the start point of (0,0). If the digit is even then it means that the matrix must be surveyed vertically and for odd numbers surveyed horizontally. This process will continue until the desired key length is being produced. For odd numbers if while key generation algorithm is picking up bits from the matrix reaches to the end of row, then the algorithm will pick up bits from beginning of next row. For even numbers if it reaches to end of column then will continue form beginning of next column. In the case of finishing 24 digits before producing the desired key length then the 24 digits will be used again and if it obtains the desired length before finishing the numbers then the remaining digits will be ignored. Fig. 8 illustrates the survey of a matrix with the first 12 digits if the digits are 493148769486.

In addition to different key generation intervals, the interpreter can produce keys of different lengths. A combination of different key lengths and three key lifetime types produces a variety of key types which vary from long minute keys to short daily keys that are convenient for different purposes. The strongest type is the long minute key which can guaranty security of top secret data encryption and the weakest type is the short daily key which is suitable for providing the lowest level of security.

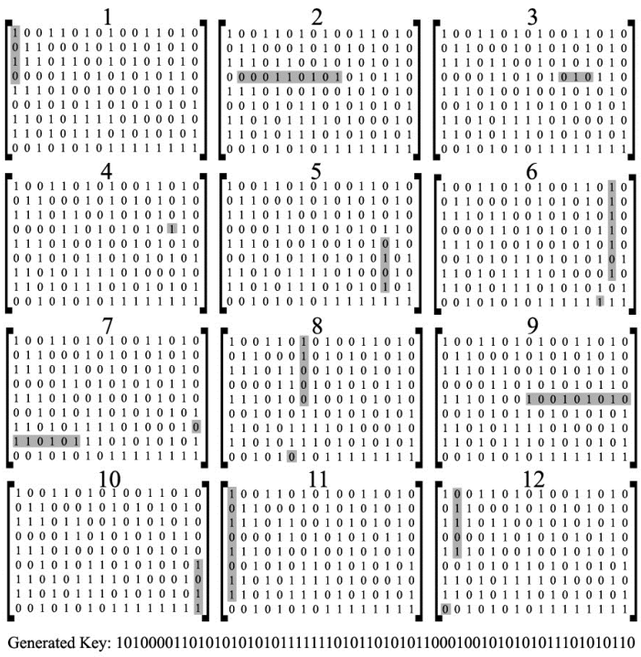


Fig. 8. Matrix technique bit-extraction illustration

3.2.6 Nodes' joining and disjoining process

For a new node joining it must send its request to the IKM server and if it passes the authentication process then it will be eligible to download the interpreter to start communicating securely with other parties. Those nodes which have already downloaded the interpreter, but for a while were inactive, must send a test packet to a neighbour node. If the test packet replied then it shows the current interpreter is still valid, but if no response is received then the process of downloading a new version is the same as joining a new node.

If a node is going to leave sessions permanently then the installed interpreter must be removed to reduce the likelihood of compromising the interpreter. If the disjoined node is trustworthy and the interpreter is removed successfully then other nodes can continue utilizing the current version, or else revoking the current version and installing the new version is a compulsory process. To renew the interpreter version, only replacing the old 24 digits with new 24 digits is enough. For example, if three counterpart companies are going to establish secure sessions then they can utilize IKM for providing security. Therefore, all computers must install the same interpreter to enable them to communicate securely. While they are working together, nodes can join and disjoin sessions, but if one of companies decides to disjoin permanently then the current interpreter must be revoked and a new version must be distributed among authorized parties.

4. Security and performance comparison between IKM and master-key

To compare two methods, their features and capabilities must be compared to help readers choose more convenient choice regarding their requirements. To do so, in this section we compare IKM and Master-Key methods in term of security and performance.

4.1 Security comparison between master-key and IKM

To compare security between master-key and IKM, five important features are analyzed and compared.

4.1.1 Key storage

One issue in key management is storing keys for future use which means encrypting the key and restoring it after going through a password check. This process increases the probability of compromising keys, but IKM only produces the key when needed and never saves it as the keys are changing continuously. Even at double valid key time after a few seconds the interpreter eliminates the old key from the memory.

4.1.2 Replacing multiple key distribution with one time interpreter distribution

Some attacks like man in the middle attack, known plain text attack, replay attack and brute force attack endanger the safety of the key distribution process (Techateerawat & Jennings, 2007). By increasing key distribution times, the likelihood of attacks happening will also increase. IKM only needs interpreter distribution to be done once and afterwards will deploy unlimited fresh keys without any necessity for key distribution. In current practices to have unique key per session for n sessions, n times key distribution is needed which n times increases the likelihood of compromising. But for IKM this process will only be run once and there is less likelihood of compromising. By reducing n times to once, compromising likelihood reduces to 1/n as well. Since IKM only once goes through key distribution process and establishes more complicated secure channels then it can provide higher level of security.

4.1.3 Utilizing unique or multiple key per session

Ideally in cryptography every transaction must be encrypted with a unique key, but in light of cost and difficulties in implementation it is not practiced widely yet. IKM nodes produce keys instead of receiving them and generating key makes less traffic than distributing it over the network. Since generating keys does not impose traffic on the network and has less cost, therefore node-side key production is cheaper and easier, and converts the ideal of cryptography into reality. For example, if every ATM user took 90 seconds to do a transaction and the ATM uses a per minute IKM key, then every ATM transaction would be encrypted with a unique key meaning the ideal encryption becomes reality.

4.1.4 Attack resistance capability

During the running time the interpreter records the amount and type of incorrect packets received and sends this to the IKM server. Depending on the received statistics the IKM server will decide whether that specific interpreter must continue working, stop temporarily or stop working completely. Also regarding received statistics the IKM server will decide whether is better to renew the current interpreter or whether it is still safe to produce a key.

The IKM structure enables resistance to some attacks that endanger the safety of key production and distribution, and reacts intelligently against attacks while there is no same feature in master-key.

4.1.5 No key revocation

Since IKM keys have a predefined validity term, they will expire automatically and there is no need for a revocation call unless for revoking the interpreter. But each key of the master-key technique must be expired by issuing a revocation call from the server. Therefore, no necessity for a key revocation is counted as an advantage of IKM versus master-key.

4.2 Performance comparison between IKM and master-key

To compare performance between IKM and master-key, imposed traffic load on network for specific duration and imposed traffic load on network per generated key are calculated and compared.

4.2.1 Key management imposed traffic load

To make an analogy about imposed key management traffic between IKM and master-key we can assume each key will last for one day and bit-stream reloading happens once a week. The contents of table 1 represent load of activities in the real world and the following formulas are used to calculate table 2 and table 3, and Fig. 9 values are used to make an analogy between these two methods. Table 3 and table 4 show differences of running master-key and IKM for minutely, hourly and daily keys.

$$\text{Master-key load for } n_w = (k_{dl} + k_{rl}) * 7 * n_w \quad (1)$$

$$\text{IKM load for } n_w = (i_{dl} + i_{rl}) + (n_w * b_{dl}) \quad (2)$$

Item explanation	Metric	Symbol
Key distribution load (master-key)	1KB	k_{dl}
Key revocation load (master-key)	0.5KB	k_{rl}
Interpreter distribution load (IKM)	50KB	i_{dl}
Interpreter revocation load (IKM)	0.5KB	i_{rl}
Bit-stream downloading load	0.5KB	b_{dl}
Number of weeks	-	n_w

Table 1. Traffic load and symbol of activities

Number of weeks	Master-key	IKM
1	10.5	51
4	42	52.5
13	136.5	57
26	273	63.5
52	546	76.5

Table 2. Daily key imposed load on server and network per node

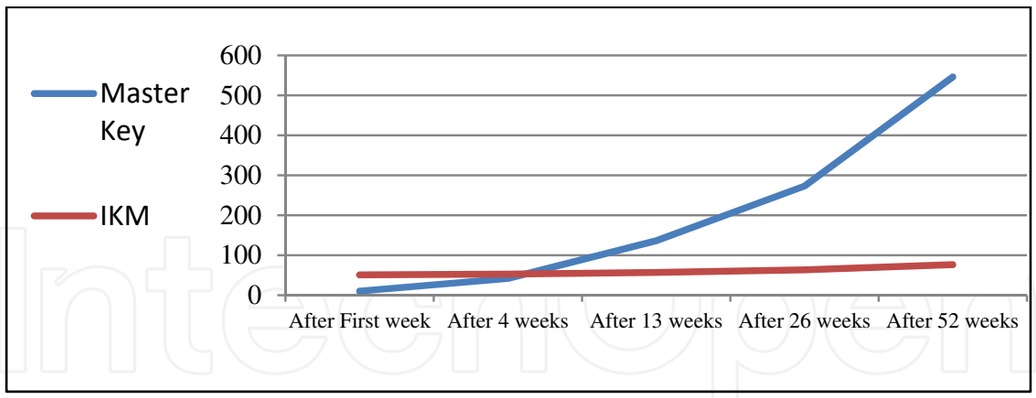


Fig. 9. Analogy of IKM and master-key traffic load on key management server for daily key

As is visible in the results of table 3 and table 4 after one year IKM's imposed traffic for the daily key is $\frac{1}{7}$, for the hourly key is $\frac{1}{171}$, and for the minutely key is $\frac{1}{10345}$ of master-key imposed traffic.

Key per session	
No. of keys	Load per key
364	1.5KB

Table 3. Number and imposed traffic load of key per session after 52 weeks

IKM					
Minutely key		Hourly key		Daily key	
No. of keys	Load per key	No. of keys	Load per key	No. of keys	Load per key
524160	0.000145KB	8736	0.00875 KB	364	0.208 KB

Table 4. Number and imposed traffic load of IKM keys after 52 weeks for minutely, hourly and daily keys

Feature	**IKM**	**Master-key**
Replay attack resistance	√	
Man in the middle resistance		√
Brute force attack resistance	√	
Less key management traffic	√	
Intelligent reaction against attackers	√	
Easy key distribution	√	
Easy key revocation	√	
Dynamism	√	
Less traffic per key	√	

Table 5. IKM and master-key features comparison

4.2.2 Analysis of comparison results

Figure 9 is drawn based on table 1 and table 2 values. It shows that employing IKM would be more beneficial if the number of needed fresh keys is over 34. When the number of fresh keys for sessions is less than 34, utilizing master-key is easier and more cost effective, but when it exceeds this threshold then IKM would be more secure and economic. As the number of needed keys increases, IKM imposes less traffic and even after one year for IKM's minutely key it would be less than one bit per generated key.

The second important point is the level of desired security. IKM reacts more intelligently than master-key against attacks. Because of IKM's structure, running brute force and replay attacks against IKM is impossible, but for man in the middle master-key is safer.

Altogether, if terms of running encrypted sessions longer than 34 keys (34 days for daily key, 34 hours for hourly key or 34 minutes for minutely key) and if a safer technique is needed then IKM is the more convenient choice, but for short-term sessions the master-key is more reasonable.

5. Conclusion

Cryptography is the technique of making information unintelligible for unauthorized parties. To do so many techniques have been developed which guaranty secrecy of encrypted information. Master-key is one internationally approved technique that can generate cryptographic keys using some environmental parameters like parties' information. Interpretative key management (IKM) framework is a newly devised method that has many similarities with master-key such as the power of key generation and using environmental parameters in key generation.

Since both techniques have some features in common, this article gives a comparative analysis to help researchers or developers who are going to utilize one of them and gives a guide for choosing between these two regarding their requirements and criteria.

A comparison of the results shows that for short-term or temporary secured sessions utilizing master-key is easier and more cost effective. But in the long run, utilizing IKM is not only more beneficial but also, because of its intelligent attack resiliency features, is more secure. Regarding imposed traffic load on the network, threshold of moving from master key to IKM is using 34 keys which depend on chosen IKM key generation method would vary from 34 minutes to 34 days. From the point of view of equipment needed both schemes need to have a server. For master-key the server is responsible of generating and distributing fresh keys and with regard to providing a key per session it is almost steadily in the process of generating and distributing fresh keys. While for IKM the server generates and distributes the interpreter once and then afterwards only supervises the security status of nodes.

To sum up the main criteria for choosing either master-key or IKM is the period of running secured sessions and the necessity for intelligent security features.

6. References

Chen, L. 2009. *Recommendation for Key Derivation Using Pseudorandom Functions,* National Institute of Standards and Technology (NIST), Retrieved from

http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf

Fumy, W.z & Landrock, P. (1993). Principles of Key Management. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS,* Vol. 11, No. 5, pp. 785-793

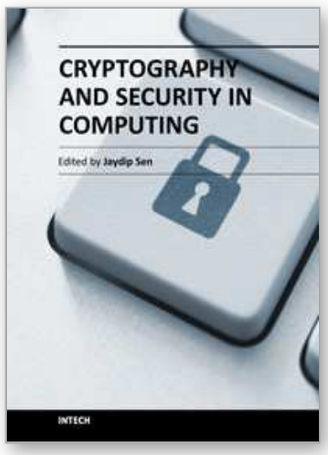
Onmez, O. 2009. *Symmetric Key Management: Key Derivation and Key Wrap*, Ruhr-Universit¨at Bochum, Retrieved from http://www.emsec.rub.de/media/crypto/attachments/files/2011/03/soenmez.pdf

Piper, F. 2002 , Murphy, S. *Cryptography: A Very Short Introduction,* Oxford University Press

Chaeikar, S. S. 2010. Interpretative Key Management (IKM), A Novel Framework, Proceedings of 2010 Second International Conference on Computer Research and Development, Kuala Lumpur, Malaysia, 2010

Chaeikar, S. S. 2010. Node Based Interpretative Key Management Framework, Proceedings of The 2010 Congress in Computer science, Computer engineering, and Applied Computing (The 2010 International Conference on Security and Management SAM'10), WORLDCOMP'2010, Las Vegas, USA, July 2010

Techateerawat, P. 2007, Analyzing the Key Distribution from Security Attacks in Wireless Sensor. Proceedings of Springer Innovative Algorithms and Techniques in Automation 2007, pp 353-357



Cryptography and Security in Computing

Edited by Dr. Jaydip Sen

ISBN 978-953-51-0179-6
Hard cover, 242 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012

The purpose of this book is to present some of the critical security challenges in today's computing world and to discuss mechanisms for defending against those attacks by using classical and modern approaches of cryptography and other defence mechanisms. It contains eleven chapters which are divided into two parts. The chapters in Part 1 of the book mostly deal with theoretical and fundamental aspects of cryptography. The chapters in Part 2, on the other hand, discuss various applications of cryptographic protocols and techniques in designing computing and network security solutions. The book will be useful for researchers, engineers, graduate and doctoral students working in cryptography and security related areas. It will also be useful for faculty members of graduate schools and universities.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen