Symmetric Key Management: Key Derivation and Key Wrap

Özlem Sönmez

11.02.2009

Seminararbeit
Ruhr-Universität Bochum

Chair for Communication Security
Prof. Dr.-Ing. Christof Paar

Contents

1 Introduction

A symmetric cryptosystem distinguishes the characteristics that the encryption key and decryption key are identical or at least in such a connection that one can be derived of the other without great effort.
Symmetric key algorithms can be divided in two groups: Stream ciphering and Block ciphering. In stream ciphers the bits of a message are encrypted one after another. In block ciphers a group of bits is encrypted at the same time (64 bits are common). AES can encrypt 128 bits.
Key derivation is very important for symmetric cryptosystems because one key can be derived of the other. Concerning key derivation at least one key has to be exchanged by the partners of communication and therefore it has to be protected. So this secret key must be sent safely before the communication can start. Key wrap can be used for encapsulating the key material.
One of the most important aspects of a cryptographical system is the key management which manages

- key generation
- key allocation and exchange
- key storage
- key adaptation

For the case that two parties share the same symmetric key often separate keys are needed for different cryptographic functions. One key can be used for an encryption algorithm and at the same time another key can be used for an integrity guarding algorithm like a message authentication code or a session key. For multiple communication partners separate keys are needed. They can be generated by a trustable party from a master key. For the derivation of such keys key derivation functions and key wrapping algorithms are needed. With the aid of examples these topics will be argued and completed with the methods of transfer.
Under the main subject Modes of Operation for Block Ciphers and Hash Functions the special subject Symmetric Key Management: Key Derivation and Key Wrap will be covered. After a definition of items like symmetric cryptography, key derivation and key wrap the main topic and the methods which are used with it will be introduced with examples.

2 What is Symmetric Cryptography?

The most important basic functions are the encryption and decryption of information. An encryption algorithm or a cipher code are mathematical functions. The security of the encryption data depends on two aspects: The strength of the encryption algorithm and confidentiality of the key.

The formal definition of an encryption algorithm (E,D,P,K,C) [Carter at al.]

- p plaintext
- K keys
- c ciphertext
- E encryption $E_K(p) = c$
- D decryption $D_K(c) = p$

The decryption function D is the inverse function of the encryption function.That means:

$$D_K(E_K(p)) = p$$

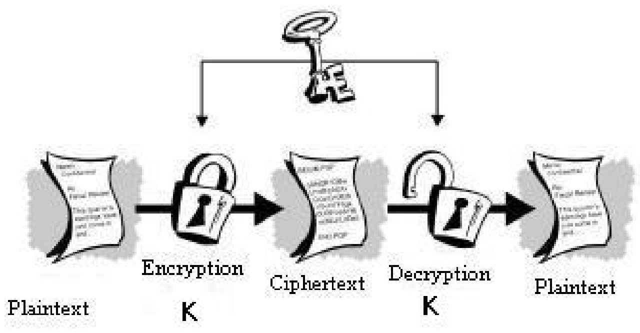

Figure 2.1: Symmetric Cryptography [Mültin]

In symmetric algorithms it is differentiated between stream ciphers, which encode the data bitwise, and block ciphers, which separate the data usually into blocks of 64 or 128 bits and each block is encrypted with the same key.

The security of that system depends on the keystreamgenerator. The greatest disadvantages of the symmetric systems are the use of the same key for encryption and decryption. That means with the encrypted information the key has to be transmitted, too [Buchmann].

3 Key Management in Symmetric Cryptography

Key management should be able to generate a secret key between two parties. The aim is to store it, to prove the authenticity of the keys and of the communicating parties. It has a high priority for lasting security of cryptographic applications. As soon as a key has been generated it must be prevented that the key is reachable for third parties. [3.]

Secure communication uses symmetric keys for pure encryption (confidentiality and integrity) and asymmetric keys for symmetric key exchange and digital signatures.

The first component of a typical key management system is building a database which is used for storing the key. There are several key classes like public, private and session keys. For each class of these keys another way of storage is needed. Key generation is the second component of a key management systems. A reliable session key has to be implemented to grant an effective encryption at the data transfer. In fact the problem at generating session keys is to find a top level of good keys. If the good session key cannot be generated, there is also no perfect key exchange method for it.

It is very important that the keys are generated randomly. For that it does not play a role if the key will be used as a symmetric or an asymmetric one.

Key exchange is third component of the key management. A secure connection requires that all communications parties have the shared key which is generated by one party and shared with the other parties securely.

The last component of key management systems is key authentication. All participants have to know if the session key they got is from the right partner. That prevents the possibility of man in the middle attack [1] [2.].

4 Pseudo Random Function

A PRF is a deterministic function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ which is efficient. Efficient means computable in polynomial time. It takes the two inputs $s, k \in \{0,1\}^n$. In this cases$s$ is a variable and k is hidden random seed. So $f(s,k) = f_k(s)$.

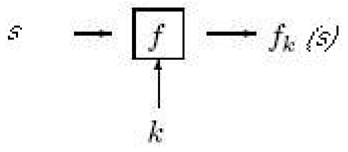

Figure 4.1: Pseudo Random Function [4.]

A real random function is rather a table with random entries. The function $s \rightarrow f_k(s)$ is considered as a good PRF when it looks like a random function. To decide its goodness the function has to be considered as a black box which computes the function $\{0,1\}^n \rightarrow \{0,1\}^n$ If it can be classified as

- a true random function, or
- a $f_k(s)$ with random k [4.]

For key derivation either the keyed hash Message Authentication Code (HMAC) or the cipher-based Message Authentication Code (CMAC) can be used.

HMAC:

- cryptographic funktion: cryptographic hashfunction like MD5 or SHA-1
- security: choice of key, secure key exchange machanisms, frequent key refreshments, good secrecy protections [Hmac.]

CMAC:

- cryptographic funktion: symmetric- key block ciphers like AES or CBC-MAC
- security: strength of symmetric- key block cipher, correctness of implementation, security and implementation of key management, strength of secret key [Cmac.]

5 Key Derivation

What is Key Derivation?

A Key Derivation Function (KDF) is a function which uses an input key and other input data to derive key material that can be used by cryptographic algorithms. The input key is called a key derivation key. It can be generated by an approved cryptographic random bit generator or by an approved automated key-establishment process [5.].
Any disjoint segments of the derived keying material (with the required lengths) can be used as cryptographic keys for the corresponding algorithm. In order to make sure that different parties will get the same keys from the derived key material, the cryptographic scheme employing a KDF must define the way to convert it into different keys. For example, when 128 bits of key material are derived, it should be specified that the first 64 bits will be used as a key for a message authentication code and the second 64 bits will be used as an encryption key for a given encryption algorithm. The key material can also be segmented into multiple keys. Depending on the intended length of the key material that has to be derived, the KDF may need multiple iterations of the pseudo random function (PRF).
The key material derived from a given key derivation key can also be used for several key derivation keys. So a key hierarchy can be established where a KDF is used with a higher-level parent key derivation key (and other input data) to derive a number of lower-level child keys (see Figure 5.1).
An improperly defined key derivation function can make the derived key material easy to attack. The key derivation function itself can achieve some security properties. One example is that the overall security of the derived key material depends on the protocols that establish the key derivation key.

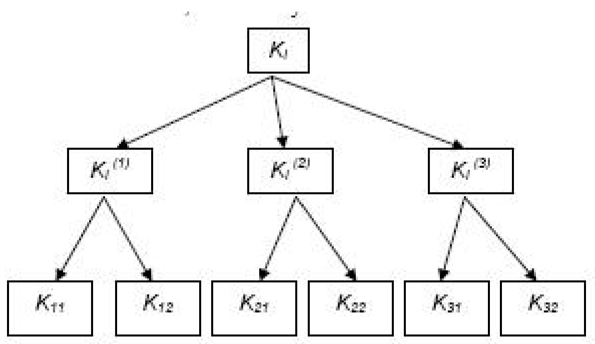

Figure 5.1: Key Hierarchy [7]

Why do we need Key Derivation?

Important points

The security strength of a key derivation function is measured by the amount of work required to distinguish the output of the KDF from a truly uniformly distributed bit string of the same length. The key derivation key K_I should the only unknown input to the KDF.

Given a set of input data and the corresponding output data the key K_I can be recovered in at most 2^w executions of the KDF through an exhaustive search over all possible K_I values.

For some KDFs the length of the key derivation key is defined by the PRF used for the derivation. When CMAC is used as a PRF the key length is uniquely determined by the block cipher. So the implementation has to check if the key derivation key length is consistent with the length required by the PRF. Some PRFs can use different key lengths. If the HMAC is used as the PRF a KDF can use a key of any length. But when it is longer than the block length of the hash function for HMAC, the key will be hashed to h bits first, where h is the length of the hash function output. In this case the hashed key can be recovered in at most 2^h computations of the PRF. That means the security strength does not increase with a longer key length.

The length L of the derived key material is dependent on the requirements of the cryptographic algorithms that rely on the KDF output. The length of a given cryptographic key is determined by the algorithm that will employ it. That means block cipher or a message authentication code and the desired security strength. So any segment of the derived key material with the required length can be used as a key. When multiple keys are obtained from the derived key material, they have to be selected from segments of the KDF output that do not overlap. Therefore the value of L should be equal to or greater than the sum of the lengths of the keys that will be obtained from the derived key material. So the derived keying material should not be used as a key stream for a stream cipher.

The input data of a key derivation function consists of different data fields like a the label, the context and the length of the output key material which are encoded as a binary string. The encryption method should define a one-to-one mapping from the set of all possible input information for that data field to a set of the corresponding binary strings. The different data fields should be assembled in a specific order. The encryption can be be defined in a larger context as the protocol that uses a key derivation function. It should be designed for explicit conversion of the combined input information to a unique binary string. This is to prevent attacks on the KDF that depend on manipulating the input data.

Some of the notations

- K_I Key derivation key,
- K_0 Key material output
- Label string that identifies the purpose zweck
- Context binary string containing the information related to the derived keying material
- IV binary string that is used as an initial value (can also be empty)
- L length (in bits) of the derived key material K_0
- h An integer that indicates the length (in bits) of the output of the PRF.
- n number of iterations of the PRF to generate L bits of key material
- w an integer with the length of a key derivation in bits.
- i a counter, a binary string of length r that is an input to each iteration of a PRF in counter mode and (optionally) in feedback mode and double-pipeline iteration mode.
- r An integer, smaller or equal to 32, that indicates the length of the binary representation of the counter i.

- $\{X\}$ Used to indicate that the data X is an optional input to the key derivation function.

- $0x00$ An all zero octet. An optional data field used to indicate a separation of different variable length data fields.

A key derivation function iterates a pseudorandom function n times and puts the the outputs together until L bits of keying material are generated.

$n = \lceil L/h \rceil$ For counter mode, n shall not be larger than 2^{r-1}, where $r \leq 32$ is the binary length of the counter.

For feedback mode and double-pipeline iteration mode, n is limited to $2^{32} - 1$ in this section based on the fact that $L = (2^{32} - 1)\, h$ bits keying material is more than enough for most applications.

For each of the iterations of the PRF, the key derivation key K_I is used as the key.

The input data consists of an iteration variable and a string of fixed input data.

Depending on the mode of iteration, the iteration variable can be a counter, the output of the PRF from the previous iteration, a combination of both, or an output from the first pipeline iteration (double-pipeline iteration mode). The length for each data field and an order can be defined explicitly [7].

Keys can also be derived from secret passwords. That is because passwords usually can not be used as cryptographic keys. There are several versions of KDF(KDF1, KDF2, KDF3, KDF4) and password-based KDF(PBKDF1, PBKDF2) [6.]

Examples for KDFs

Algorithm KDF3

[6.]

The Input is Z (shared secret) string in byte and the Hash is the hash function which produces the Output as *hLen* in bytes. *kLen* is the indented length of the key material. *pAmt* is an integer that has to be 4 or more than 4 . There is also a variable that is called *[OtherInfo]* , which is optional. As the Output we get derived keyK_I ,K with the length *kLen* in bytes.

Algorithm 5.0.1 .

```
1.Set d = ceiling(kLen/hLen).
2.Set T = "..", the empty string.
3.for Counter = 0 to d-1 do:
C = IntegerToString(Counter, pAmt)
$T = T||Hash\,(C||Z||\,[OtherInfo])$
4.Output the first kLen bytes of T as $K_I$.
```

Algorithm PBKDF1

[6.]

Length of the derived key$K_I \leq$ length of hash function (MD5-16 byte and SHA-1 20 byte) The Input are P (password) string in byte , S (salt) 8-byte sitring and C (a positive integer). The Hash is the hash function which produces the Output as *hLen* in bytes. *kLen* is the indented length of the key material. There is also a variable that is called *[OtherInfo]* , which is optional. As the Output we get derived keyK_I ,K with the length *kLen* in bytes.

Algorithm 5.0.2 .

```
1.If kLen >hLen then stop with error "kLen is too long"
2.T1 = $Hash(P||S)$
for $i = 2$ to $C$
$Ti = Hash\,(Ti - 1)$
3.Output the first kLen bytes of TC as $K_I$.
```

6 Key Wrap

Why Key Wrap?

Key Wrap is a classification of symmetric encryption algorithms and it is designed to encrypt cryptographic key material. It is developed for generating

- protective keys for untrusting storage
- transmitting keys for untrusting communications networks.

Key wraps are constructed from block ciphers and cryptographic hash functions [,]8.

What is Key Wrap?

Key Wrap with AES

Key Wrap can e.g. use the Advanced Encryption Standard (AES) as a primitive to securely encrypt plaintext key(s) with any associated integrity information and data. The combination can be longer than the width of the AES blocksize which consists of 128-bits. Each ciphertext bit is a highly non-linear function and that is why unwrapping each plaintext bit should be a highly nonlinear function of it. But it sufficient to approximate an ideal pseudo random permutation to such a degree that exploiting of unwanted doings and guessing the AES engine key is impossible. The key wrap algorithm is an algorithm to wrap keys and other input data together.
The key wrap operates on blocks of 64bits. Before being wrapped, the key data which is longer, is parsed into n blocks of 64 bits. The only restriction is that the key wrap algorithm places on n is that n be at least two. It is recognized that $n \leq 4$ can host all supported AES key sizes.
The AES key wrap can be configured to use any of the three key sizes supported by AES. The choice of a key size effects the overall security provided by the key wrap, but it does not alter the the key wrap algorithm. The key wrap will be described generally hereafter (the Key Encryption Key KEK can have a length of $128-$ bits, 192 bits or 256 bits).

Key Data Integrity with the Initial Value
The initial value (IV) refers to the value assigned to A^0 in the first step of the

wrapping process. This value is used for an integrity check on the key data. In the final step of the unwrapping process, the recovered value of A^0 is compared to the expected value of A^0. If there is a match, the key is accepted as valid and is returned by the unwrapping algorithm. If it does not match, the key is not accepted as valid and the unwrapping algorithm returns an error. The exact properties achieved by this integrity check depend on the definition of IV. Keys can to be checked for their integrity throughout their lifecycles or just when they are unwrapped.
When the key wrap is used as part of a larger key management protocol or system, the desired scope for data integrity can be more than just the key data or the desired lifecycle. For such problems alternative definitions of the initial value can be used [9.].

Example

Key Wrap [9.]

The inputs to the key wrapping process are the KEK and the plaintext that has to be wrapped. The plaintext consists of n 64-bit blocks, containing the key data being wrapped (see Figure 6.1).

Inputs:
Plaintext, n64-bit values $\{P_1, P_2, ..., P_n\}$,
Key, K (the KEK).
Outputs:
Ciphertext, $(n+1)\,64-$bit values $\{C_1, C_2, ..., C_n\}$

Algorithm 6.0.3 .

```
1. Initialize variables
Set $A^0 = IV$, and initial value $s = 6n$
For $i = 1, ....., n$,
$R_i^0 = P_i$

2. Calculate intermediate values
For $t = 1, ...s$, where $s = 6n$
$A^t = MSB_64\left(AES_K\left(A^{t-1} \mid R_1^{t-1}\right)\right) \oplus t$
For $i = 1, ...., n-1$
$R_i^t = R_{i+1}^{t-1}$
$R_t^n = LSB_{64}\left(AES_K\left(A^{t-1} \mid R_1^{t-1}\right)\right)$

3. Output the results
Set $C_0 = A^t$
```

```
For $i = 1, ...., n$
$C_i = R_i^t$
```

The key Unwrap

The inputs to the unwrap process are the KEK and $(n+1)\,64-$bit blocks of ciphertext consisting of previously wrapped key. It returns n blocks of plaintext consisting of the $n64-$bit blocks of the decrypted key data(see Figure 6.2).

Inputs:
Ciphertext $(n+1)\,64-$bit values $\{C_1, C_2, ..., C_n\}$
Key, K (the KEK)
Outputs: Plaintext$n64-$bit values $\{P_1, P_2, ..., P_n\}$

Algorithm 6.0.4 .

```
1. Initialize variables
Set $A^s = C_0$ where $s = 6n$
For $i = 1, ...., n$
$R_i^s = C_i$

2.Calculate the intermediate values
For $t = s, ..., 1$
$A^{t-1} = MSB_{64}\left(AES_K^{-1}\left((A^t \oplus t) \mid R_n^t\right)\right)$
$R_1^{t-1} = LSB_{64}\left(AES_K^{-1}\left((A^t \oplus t) \mid R_n^t\right)\right)$
For $i = 2, .., n$
$R_i^{t-1} = R_{i-1}^t$

3.Output the results
If $A_0$ is an appropriate IV
Then
For $i = 1, ...., n$
$P_i = R_i^0$
Else
Return an error
```

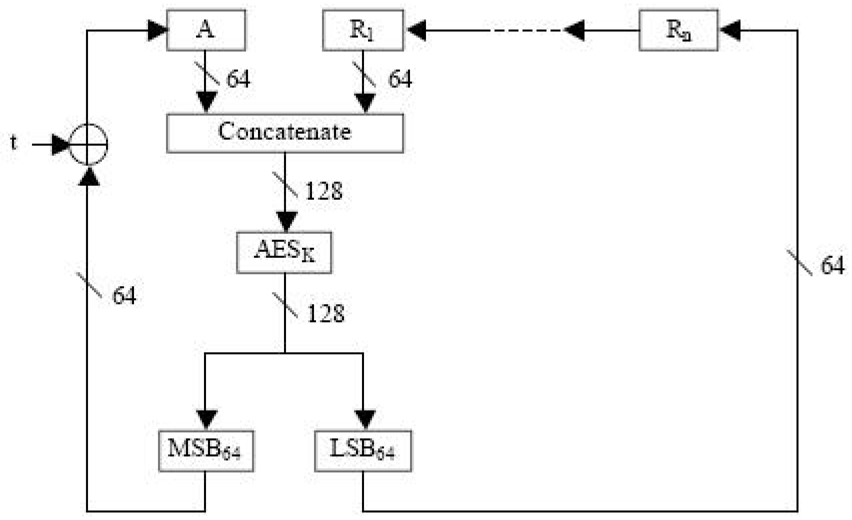

Figure 6.1: The motion of the key wrap [9.]

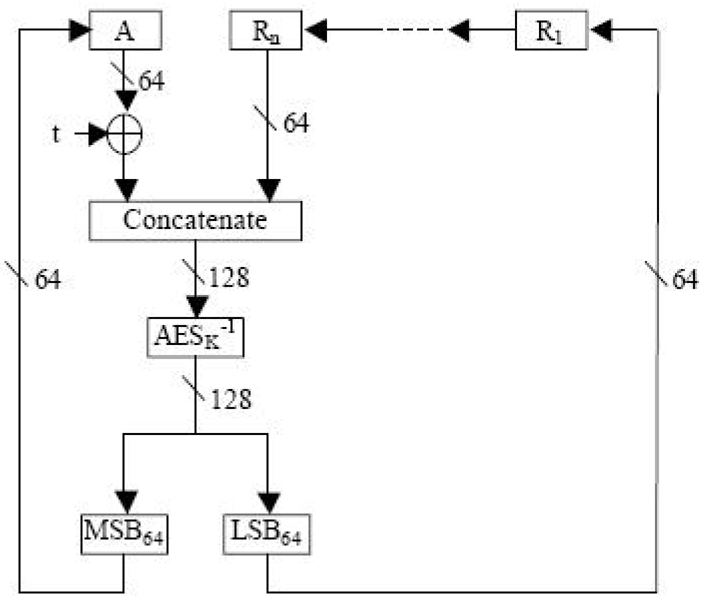

Figure 6.2: The motion of the key unwrap [9.]

Bibliography

...[Buchmann] Johannes Buchmann *Einführung in die Kryptographie.* Berlin Springer-Verlag 2001 3.erweiterte Auflage 4. Chapter p.59

...[Mültin] *http://www.ipd.uka.de/ oosem/SecIS06/Ausarbeitungen/Seminarausarbeitung-Mueltin-Kryptographie.pdf.*

...[Carter at al.] *http://briancarter.info/pubs/symmetric_cryptosystems_and_symmetric_key_m*
Brian A. Carter, Ari Kassin and Tanja Magoc,Symmetric Cryptosystems and Symmetric Key Management

...[1] Ralf Spenneberg *VPNmit Linux Grundlagen und Anwendung Virtueller Privater Netzwerke mit Open Source Tools.* Addison-Wesley-Verlag 2003 4.Chapter p.109

...[2] *www.cryptoshop.com.*
Knowledge-Base,Kryptographie,Schlsselmanagement

...[3] *http://en.wikipedia.org/wiki/Key_management.*

...[4] *http://www.nada.kth.se/kurser/kth/2D1441/semteo03/lecturenotes/prf.pdf..*

...[Cmac] *http://tools.ietf.org/html/draftsongleeaescmac02.*

...[Hmac] *http://tools.ietf.org/html/rfc2104.*

...[5] *http://en.wikipedia.org/wiki/Key_derivation_function.*

...[6] *http://www.di-mgt.com.au/cryptoKDFs.htmlPKCS5.*

...[7]*http://csrc.nist.gov/publications/nistpubs/80056A/SP800-56A_Revision1_Mar082007.pdf.*
Recommendation for Key Derivation Using Pseudorandom Functions

...[8] *http://en.wikipedia.org/wiki/Key_Wrap.*

...[9]*http://csrc.nist.gov/groups/ST/toolkit/documents/kms/key-wrap.pdf.* AES Key Wrap specification