# Background ON Cryptographic algorithms

To establish a secure system, cryptographic primitive algorithms are employed. A brief explanation and introduction for cryptographic primitives are provided in this thesis to provide unity in the document.

In following sections hash functions, hash chains and HMAC functions are explained. Moreover symmetric cryptography is described. Finally explanation for public key cryptography is provided at the end of this section.

## 1.1. Hash Functions

Hash functions [1] are irreversible mathematical functions that map input strings of variable length to fixed sized output strings. Hash functions are usually employed for improving time performance of table lookup or data comparison tasks such as finding items in a database, discovering repeated or analogous records in a bulky file, finding similar springs in a DNA string and cryptographic purposes.



Figure 1. Hash Function Example [1]

Figure depicts the hash function flow. The function maps a longer message to a 160-bit bit string. The output length depends on the hash algorithm; various hash algorithms have different output sizes.

Hash functions could receive various sized parameters but generate fixed sized input strings. Compared to mainstream cryptographic algorithms, hash functions are fairly cost-effective in both power and time consumption. Light-weightiness of hash functions make them eligible for security systems.

A hash function should satisfy following properties:

1. Given a message m, the message digest h(m) can be calculated very quickly.
2. Given a y, it is computationally infeasible to find an ~1 with h (m'). =*Y* (in other words, *h* is a one-way, or collision resistant, function)

## 1.2 Hash Chains

Applying a hash algorithm to an initial value and using the output as an input for the next hash function forms a hash chain. Every output of a hash algorithm represents a link in the chain. Length of the hash chain is determined by the number of times the hash algorithm is executed.



Figure 2. Hash Chain Depiction and Usage [2]

Since hash functions are irreversible, it is easy to go forward in the chain but it is not computationally feasible to go backwards. Which means a person could find any value on the chain if she knows the initial value but this situation is not possible for a person who knows the last value on the chain.

A hash chain with 5 elements is denoted as:

More generally, hash chain with *n* elements is denoted as:

n times

Because of the fact that hash functions are one-way mathematical functions, it is appropriate to say that hash functions are good tools for security systems communicating through insecure links. Knowing the first link in the chain gives the opportunity to verify the following links in the chain as well. If one could establish a system, successful at distributing the first link in the chain, it is feasible to use hash chains as future keys or secrets for other cryptographic functions.

Hash chains are easy-to-deploy and cost-effective therefore they are widely used in cryptographic systems. Especially for systems that have delicacy for computational delay hash chains are effective tools.

## 1.3 HMAC Functions

One of the main research areas in cryptography and computer networking is providing integrity and reliability on a transmitted or stored data. Message Authentication Codes (MACs) are use shared secret keys; therefore they are suitable for such integrity checks.

Classically, MACs are used between two parties that share a secret key in order to authenticate the transmitted or stored data between these parties. This protocol executes a MAC that uses a cryptographic hash function union with a secret key.

HMACs are used together with widely accepted hash functions. HMAC employs a secret key for generation and verification of the MACs. The aims of HMAC construction [3] are:

1. Using hash functions without making any changes on them. Previously implemented codes and hardware shall work with the deployment of HMAC.
2. Maintaining the original fastness of the hash functions.
3. Using and handling secret keys in a cost-effective way.
4. Providing provable and reasonable cryptographic analyzes using the previously done performance analysis of underlying hash functions.
5. Achieving faster and more robust performances in a case of a faster hash function is invented in the future. Replacement should be easy-to-achieve.

Table I. explains the parameters HMAC uses.

Table 1.1. HMAC Parameters [3]

|  |  |
| --- | --- |
| B | Block size in bytes |
| H | A secure and fast hash function |
| ipad | Inner pad, the byte x36 B times |
| K | Shared secret key |
| K0 | The key K before any process to make it B bytes long |
| L | Block size of the output of the hash function, in bytes |
| opad | Outer pad, the byte x5c repeated B times |
| t | The number of bytes of MAC. |
| text | The data that used to calculate HMAC |
| xN | Hexadecimal notation where each string N represents 4 binary bits |
|  | Exclusive-Or operation |
|  | Concatenation |

Figure 3. shows the steps of HMAC.



Figure 3. Steps of HMAC [3]

## 1.4 Symmetric Cryptography

Symmetric cryptography is the oldest kind of cryptographic primitive. This primitive employs shared secret keys between two parties. The security level of a symmetric cryptographic algorithm mostly depends on key size. Modern algorithms use at least 128-bit long keys.

Modern

# References

1. Trappe, W. Washington, L. C. Introduction to Cryptography with Coding Theory, Second Edition, pp. 218-220
2. Oligeri, G. Chessa, S. Giunta, G. Loss Tolerant Video Streaming Authentication in Heterogeneous Wireless Networks, Computer Communications, 34(11): 1307-1315, 2011.
3. American Bankers Association, *Keyed Hash Message Authentication Code*, ANSI X9.71, Washington, D.C., 2000
4. H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, Internet Engineering Task Force, Request for Comments (RFC) 2104, and February 1997.