NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

LAB 10

LAB 10: Write a Program to implement Cryptographic Hash Functions and Applications (HMAC): to understand the need, design, and applications of collision-resistant hash functions.

ROLL NO	52
NAME	Sarvesh Patil
CLASS	D15A
SUBJECT	Internet Security Lab
LO MAPPED	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers. LO2: Demonstrate Key management, distribution, and user authentication.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

AIM:
Write a Program to implement Cryptographic Hash Functions and Applications (HMAC): to understand the need, design, and applications of collision-resistant hash functions.

INTRODUCTION:
Data integrity is vital for securing communications and is usually achieved through a mechanism known as HMAC.

Overview
Data integrity checks are vital to secure communications. They enable communicating parties to verify the integrity and authenticity of the messages they receive. In secure file transfer protocols like FTPS, SFTP, and HTTPS, data integrity/message authentication is usually achieved through a mechanism known as HMAC. In this post, we explain what HMAC is, its basic inner workings, and how it secures data transfers.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Importance Of Data Integrity Checks In Secure File Transfers

Business decisions and processes are highly dependent on accurate and reliable data. If data gets tampered with and these changes go unnoticed, it could affect decisions and processes down the line. So if your data has to be transmitted over a network, especially one as perilous as the Internet, you have to take precautionary measures to preserve its integrity or at least know if it has been hacked or altered.

This is precisely the reason why secure file transfer protocols like FTPS, SFTP, and HTTPS are equipped with mechanisms for preventing threats to data integrity. The most commonly used mechanism today is HMAC.

What Is HMAC?

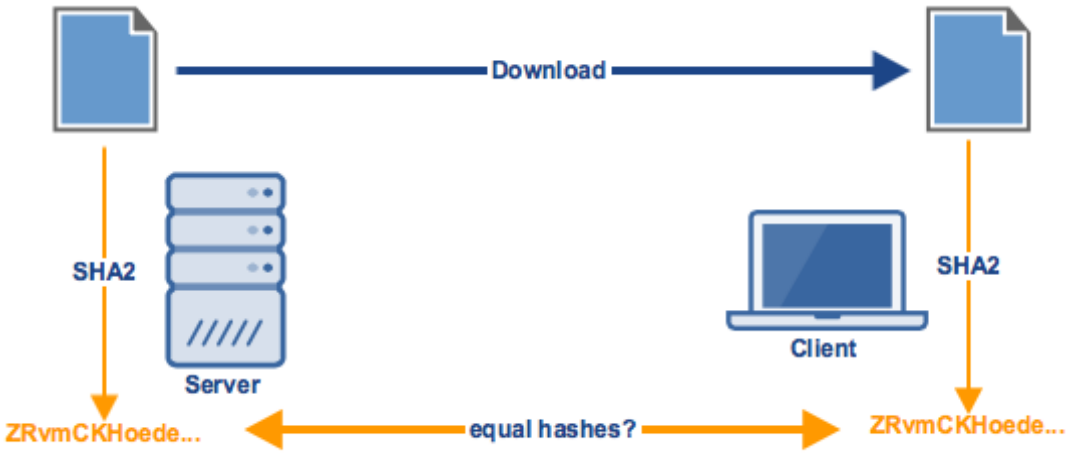
HMAC stands for Keyed-Hashing for Message Authentication. It's a message authentication code obtained by running a cryptographic hash function (like MD5, SHA1, and SHA256) over the data (to be authenticated) and a shared secret key. HMAC is specified in RFC 2104.

HMACs are almost similar to digital signatures. They both enforce integrity and authenticity. They both use cryptography keys. And they both employ hash functions. The main difference is that digital signatures use asymmetric keys, while HMACs use symmetric keys (no public key). Recommended read: Symmetric vs Asymmetric Encryption

How HMAC Works

To understand how HMAC works, let's first examine how a hashed function (on its own) could be used for conducting a data integrity check on a file transfer. Let's say a client application downloads a file from a remote server. It's assumed that the client and server have already agreed on a common hash function, for example, SHA2.

Before the server sends out the file, it first obtains a hash of that file using the SHA2 hash function. It then sends that hash (ex. a message digest) along with the file itself. Upon receiving the two items (ex. the downloaded file and the hash), the client obtains the SHA2 hash of the downloaded file and then compares it with the downloaded hash. If the two matches, then that would mean the file was not tampered with.



NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

If an attacker manages to intercept the downloaded file, alter the file's contents, and then forward the altered file to the recipient, that malicious act won't go unnoticed. That's because once the client runs the tampered file through the agreed hash algorithm, the resulting hash won't match the downloaded hash. This will let the receiver know the file was tampered with during transmission.

So a hash function should protect your files, right? Not so fast. While a hash function can establish data integrity, it can't establish authenticity. How would the client know the message it received came from a legitimate source?

That's why secure file transfer protocols like FTPS, SFTP, and HTTPS use HMACs instead of just hash functions. When two parties exchange messages through those secure file transfer protocols, those messages will be accompanied by HMACs instead of plain hashes. An HMAC employs both a hash function and a shared secret key.

A shared secret key provides exchanging parties a way to establish the authenticity of the message. That is, it provides the two parties a way of verifying whether both the message and MAC (more specifically, an HMAC) they receive really came from the party they're supposed to be transacting with.

The secret key enables this capability because it's generated during the key exchange, a preliminary process that requires the participation of the two parties. Only those two parties who participated in the key exchange would know what the shared secret key is. In turn, they would be the only ones who would be able to arrive at the same result if they compute the message's corresponding MAC using the shared secret key.

Why Is HMAC Suitable For File Transfers?

Aside from its ability to enable data integrity and message authentication, another reason why HMAC is an excellent file transfer data integrity-checking mechanism is its efficiency. As discussed in the article Understanding Hashing, hash functions can take a message of arbitrary length and transform it into a fixed-length digest. That means, even if you have relatively long messages, their corresponding message digests can remain short, allowing you to maximize bandwidth.

Choosing An HMAC Function

Because an HMAC's properties (especially its crypto strength) are highly dependent on its underlying hash function, a particular HMAC is usually identified based on that hash function. So we have HMAC algorithms that go by the names of HMAC-MD5, HMAC-SHA1, or HMAC-SHA256. You've probably heard about the collision-related vulnerabilities of MD5. It's worth noting that HMAC-MD5, in spite of its underlying MD5 hash function, isn't as affected by those vulnerabilities. Regardless, SHA-1 is still cryptographically stronger than MD5 and SHA-2 (and its different forms, like SHA-224, SHA-256, and SHA-512) is likewise cryptographically stronger than SHA1, so you will want to take that into consideration.

So which HMAC should you use? You would normally choose an HMAC based on its underlying hash function. So, for example, you would want to use HMAC-MD5 if performance is more critical to you than security. On the other hand, if security is more critical, then you might want to use HMAC-SHA256 instead.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Applications:

- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for the Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest. That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

Advantages:

HMACs are ideal for high-performance systems like routers due to the use of hash functions which are calculated and verified quickly unlike the public key systems.
Digital signatures are larger than HMACs, yet the HMACs provide comparably higher security.
HMACs are used in administrations where public key systems are prohibited.

Disadvantages:

HMACs use a shared key which may lead to non-repudiation. If either sender or receiver's key is compromised then it will be easy for attackers to create unauthorized messages.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

METHOD:

Plaintext: 1100000000111100101010

Divide the plaintext into k chunks of 8 bits.
If the kth chunk's bits are less than 8, make it 8 by padding 0's at the end.

M1= 11000000
M2= 00111100
M3= 10101000

|| denotes concatenation

M=M1||M2||M3.....Mk = 110000000011110010101000
L= Length of M = 24 in decimal = 00011000

A= Key XOR ipad = 10000101 XOR 01011100 = 11011001

B= KEY XOR opad = 10000101 XOR 00110110 = 10110011

Z0=IV||A = 1100110011011001
Z1=Hashed value of Z0||M1 = 0000011111000000
Z2=Hashed value of Z1||M2 =0010000000111100
Z3=Hashed value of Z2||M3 =0100000010101000
Z(k+1)= Hashed value of Zk||L =1000111000011000

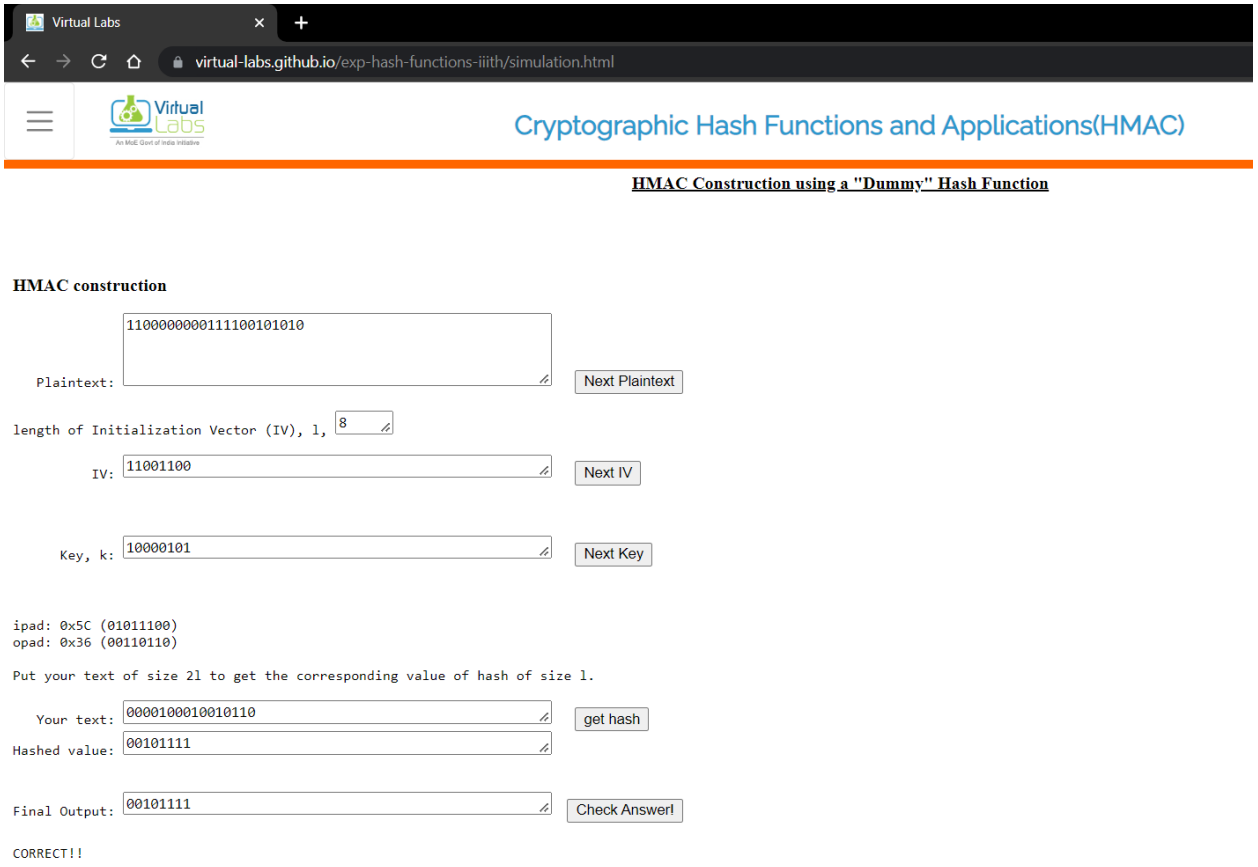
Hashed value of Z(k+1) = 10010110

P= IV||B = 1100110010110011
Q=Hashed value of P = 00001000
R= Q||Hashed value of Z(k+1) = 0000100010010110

T= Hashed value of R = Final Output = 00101111

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

RESULTS:



CONCLUSION:

We have successfully implemented HMAC using virtual labs and understood the applications of HMAC to understand the need, design and applications of collision-resistant hash functions.