PDH Academy

Detection of Errors in Digital Systems

Mark A. Strain, P.E.

PDH303

2 Hours

Course Material and Final Exam

Introduction

We are exposed to a lot of information every day, from viewing content on a website on the Internet to listening to a song on our smart phone. This information (or data) is constantly stored, transmitted and processed. It is important that the data is correct or relatively error-free. Some amount of error is acceptable, depending on the application. For example, a few bit errors in a music data file in an MP3 format are acceptable, but a few bit errors in the data being transferred to the flight controls of a rocket could be catastrophic.

Information theory is a branch of applied mathematics and electrical engineering involving the quantification of information. Claude Elwood Shannon ushered in the modern electronic communications age by developing information theory in 1948. He laid the groundwork for both the computer industry and telecommunications. A key feature of Shannon's theory was the introduction of the concept of entropy to information, which he demonstrated to be equivalent to a shortage in the information content in a message. The entropy or shortage of information in a message introduces errors in the data.

Error Detection Techniques

Most communications systems, whether wired or wireless, are unreliable to some degree. The application of error detection and correction techniques ensures the reliable delivery of data transmitted over a communication channel. Error detection also allows a user to reliably recover data stored in a memory device.

Errors are often times introduced in the channel during the transmission of data. Communication channels are subject to channel noise. Employing an error detection technique allows a user to detect an error at the receiving end. Employing an error correction technique allows a user to reconstruct (or request a retransmission of) data that was lost or corrupted. Data does not have to be transmitted great distances to benefit from error detection/correction methods. The data could be from one memory device to another. The channel could be as long as a ground station to satellite link or as short as a microprocessor to a RAM device on the same circuit board.

The basic approach of error detection and correction is to add some overhead (extra bits) to the data that will be utilized at the data's destination to determine if any errors had occurred in the channel during the transfer. The extra data (or overhead) may be inserted at regular intervals in the data stream (like a parity bit) or at the end of a packet of data or at the end of the entire transfer (like a checksum, cyclic redundancy check or hash function).

Parity

In mathematics, parity refers to the property of an integer with respect to being odd or even. In digital communications, parity refers to the number of bits within a word (e.g. a byte) being either odd or even. The property of a group of bits being odd or even is easily computed using an XOR (^) function. Therefore parity is a common method and one of the simplest forms of error detection code.

Hex Number	Binary Equivalent	Parity Computation	Parity
0xCE	1100 1110	1 ^ 1 ^ 0 ^ 0 ^ 1 ^ 1 ^ 1 ^ 0 = 1	odd parity
0xCF	1100 1111	1 ^ 1 ^ 0 ^ 0 ^ 1 ^ 1 ^ 1 ^ 1 = 0	even parity
0x58	0101 1000	0 ^ 1 ^ 0 ^ 1 ^ 1 ^ 0 ^ 0 ^ 0 = 1	odd parity
0x59	0101 1001	0 ^ 1 ^ 0 ^ 1 ^ 1 ^ 0 ^ 0 ^ 1 = 0	even parity

Table 1 - Computation of Odd and Even Parity

A parity bit is a bit that is added to a set of bits to ensure that the number of bits in the set is odd or even. There are two types of parity checks: even parity and odd parity.

Even Parity

When even parity is used, the parity bit is set to make the number of bits in a set even. If the number of bits in the set was odd the parity bit would be set to 1 to make the number even again. If the number of bits in the set was even, the parity bit would be set to 0 to maintain the even number of bits.

Odd Parity

When odd parity is used, the parity bit is set to make the number of bits in a set odd. If the number of bits in the set was even the parity bit would be set to 1 to make the number odd again. If the number of bits in the set was odd, the parity bit would be set to 0 to maintain the odd number of bits.

Hex Number	Binary Equivalent	Even Parity	Odd Parity
0x00	0000000	[0]0000000	[1]0000000
0x08	0001000	[1]0001000	[0]0001000
0x28	0101000	[0]0101000	[1]0101000
0x2A	0101010	[1]0101010	[0]0101010
0x6D	1101101	[1]1101101	[0]1101101
0x7F	1111111	[1]1111111	[0]1111111

Table 2 - Even Parity and Odd Parity

Parity may be computed in software, but often times the operation is performed by hardware because the hardware computation is faster. The parity bit may be computed and injected into the data stream by the hardware itself, like a UART (universal asynchronous receiver/transmitter). In a digital communication system the transmitter hardware (e.g. a UART) will load a set of bits into a register and compute the parity bit and insert the parity bit either as the most significant bit or least significant bit in the set. The set of bits including the parity bit will then be shifted out and transmitted.

On the other end the receiver will receive each set of bits, strip off the parity bit and compute the parity for the set of bits it received. It will compare the newly computed parity bit to the one stripped off from the receiver. If they match then the data received is deemed good. If they do not match, then an error has occurred during the transmission.

…[1] 0 0 0 1 0 0 0 [0] 0 1 0 1 0 0 0 [1] 0 1 0 1 0 1 0 [1] 1 1 0 1 1 0 1…

Figure 1 - Example Data Stream Incorporating Even Parity

…[0] 0 0 0 1 0 0 0 [1] 0 1 0 1 0 0 0 [0] 0 1 0 1 0 1 0 [0] 1 1 0 1 1 0 1…

Figure 2 - Example Data Stream Incorporating Odd Parity

The parity check is suitable only for detecting errors, not for correcting them. If a discrepancy in the parity check occurs (meaning if an error is detected), that portion of data must be discarded and retransmitted. If during a transmission the parity matches and the receiver deems the data is good, it may still be in error if two bits flipped instead of one, but there will be no way to tell from the parity check.

Transmitted data
…[1] 0 0 0 1 0 0 0 [0] 0 1 0 1 0 0 0 [1] 0 1 0 1 0 1 0 [1] 1 1 0 1 1 0 1…

Received data
…[1] 0 0 0 1 0 0 0 [0] 0 1 0 1 0 0 0 [1] 0 1 1 1 0 1 0 [1] 1 1 0 1 1 0 1…
↑
error detected (parity computes to 0, but parity received is 1)

Figure 3 - Example of Data Transmission and Error Detection

Advantages and Disadvantages of the Parity Method

Parity checking is simple, fast and takes little overhead, but is not the most robust form of error detection.

Checksum

Another form of error detection in digital systems incorporates a checksum within the data. A checksum is a fixed-sized (usually 8-bit, 16-bit or 32-bit) numerical data value that is computed over a message or block of data using some form of checksum function. Computing a checksum for a block of data and inserting the checksum at the end of the block of data before the data is transmitted is a simple error detection technique.

The integrity of the transferred data can be checked by computing the checksum over the received message and comparing it to the received checksum. The checksum function (or checksum algorithm) computes the checksum value over a block of data.

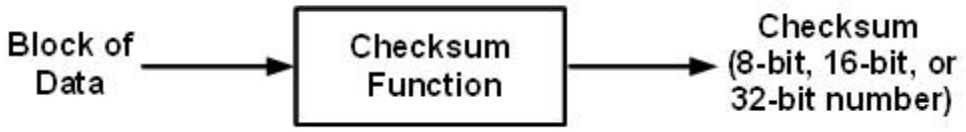


Figure 4 - Checksum Function

One method of computing a checksum is to XOR all of the words of the message block together. A word will typically be 8-bits, 16-bits or 32-bits in length. So, for a 32-bit checksum, a running XOR of every 4 bytes will be computed for the entire message. This XOR method is called a longitudinal parity check or horizontal parity check.

0xAE ^ 0xCA ^ 0x85 ^ 0x4D = 0xAC

Figure 5 - Longitudinal Parity Check Checksum

A simple form of checksum algorithm is to sum all of the words of a message in a counter. Any overflow is discarded. The counter is the same size as the words being counted.

0xAE + 0xCA + 0x85 + 0x4D = 0x4A

Figure 6 - Running Summation Checksum

During a data transmission a block of data is run through the checksum function to produce a checksum value. This value is appended to the message and the data is transmitted. The receiver receives and stores the message and strips off the checksum. The received data minus the checksum is run through the checksum function to produce a new checksum. This newly-computed checksum is compared with the received checksum. If the two match, the received data is deemed good. If the two checksums are different, an error has occurred and the data will need to be retransmitted.

Figure 7 - Packet of Data Containing 8-bit Checksum

Advantages and Disadvantages of the Checksum Method

The use of checksums as an error detection technique is simple, fast and takes little overhead in resources and time. However, the technique is weak regarding error detection. If the bytes were transmitted in a different order the checksum result would be the same. If the bytes were transposed from big endian to little endian, for example, the checksum result would be the same.

Cyclic Redundancy Check

A cyclic redundancy check (or CRC) is a technique for detecting errors in digital data. This method is sometimes called a checksum, but is more robust than the simple method of adding or XORing the words. The value computed by the CRC function is called a check value, checksum or CRC.

The technique of error detection is the same as that of the checksum method. A given block of data is run through an algorithm to compute a numerical value. The numerical value (or CRC) is appended to the block of data and the data is transmitted. The receiving end computes the CRC and compares it to the one that was received. If they match the received data is deemed good. If they differ then an error has occurred and the data will have to be retransmitted.

CRC Algorithm

The CRC algorithm is based on cyclic codes which involve polynomial arithmetic in something called a Galois field (GF). The arithmetic is done using modulo-2, which means the arithmetic is accomplished using an XOR operator.

To compute a CRC a binary message is divided by a predetermined polynomial called a generator polynomial. The remainder of this operation is the CRC.

The CRC is computed using a generator polynomial, for example, $x^8 + x^2 + x + 1$ is a generator polynomial that may be used to generate a CRC-8 (8 bits in length). The generator polynomial is selected to maximize error detecting capabilities. The length (or exponential degree) of the polynomial determines the length of the CRC. They are typically 8, 16, 32 and 64 bits in length.

A CRC may be computed in hardware or software.

Hardware Implementation

A hardware realization of a CRC circuit includes a shift register and some XOR gates. A circuit to compute the CRC for a generator polynomial of $G = x^3 + x^2 + 1$ is shown in Figure 8.

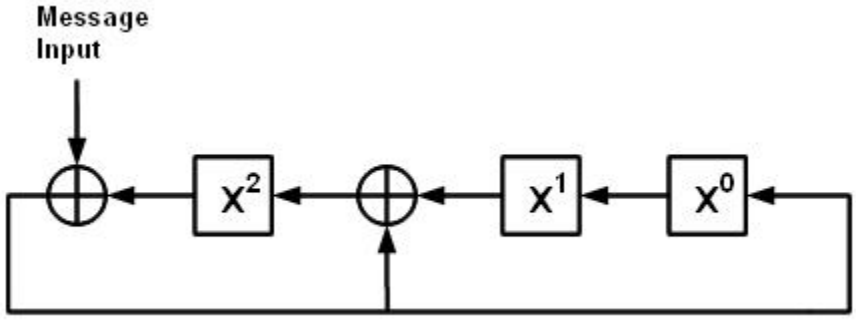


Figure 8 - Circuit of CRC Generation Using $G = x^3 + x^2 + 1$

For example, the message (M) 11010111 processed through the circuit produces a CRC of 001:

Message	CRC Register

	0 0 0
1	1 0 1
1	0 1 0
0	1 0 0
1	0 0 0
0	0 0 0
1	1 0 1
1	0 1 0
1	0 0 1

Table 3 - CRC Computed Using G = $x^3 + x^2 + 1$ and M = 11010111

So, the CRC-3 for the message M = 11010111 using the generator polynomial G = $x^3 + x^2 + 1$ is what is contained in the CRC register after the last message bit is processed: CRC-3 = 001.

A circuit used to compute the CRC for a generator polynomial of G = $x^8 + x^2 + x + 1$ is shown in Figure 9.

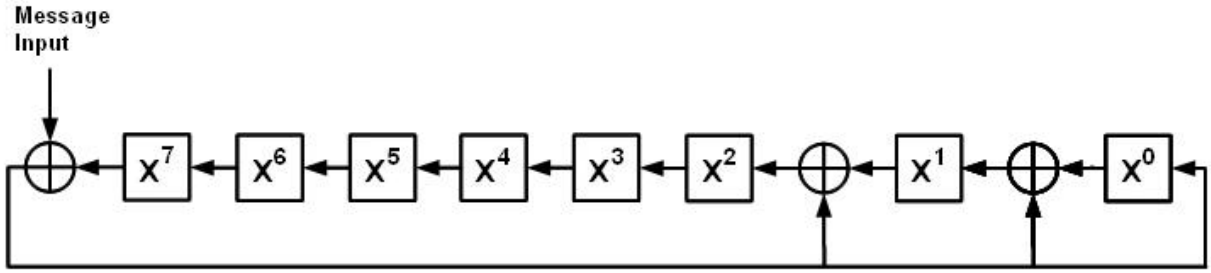


Figure 9 - Circuit of CRC Generation Using G = $x^8 + x^2 + x + 1$

For example, the message (M) 0101101011000110 (0x5AC6) processed through the circuit produces a CRC of 11010010 (0xD2):

Message	CRC Register
	0000 0000
0	0000 0000
1	0000 0111
0	0000 1110

1	0001 1011
1	0011 0001
0	0110 0010
1	1100 0011
0	1000 0001
1	0000 0010
1	0000 0011
0	0000 0110
0	0000 1100
0	0001 1000
1	0011 0111
1	0110 1001
0	1101 0010

Table 4 - CRC Computed Using $G = x^8 + x^2 + x + 1$ and M = 0101 1010 1100 0110

Software Implementation

There are several CRC algorithms written in the C programming language as well as C++ and Java available on the Internet. Some implementation methods are more efficient than others. Typical microprocessors do not have registers long enough to hold the very long bit sequences typical with modulo-2 division. Therefore, the bit sequences are usually handled byte at a time. The most inefficient software implementation is one that requires multiple statements to be executed for each bit in the message.

A more efficient software implementation is one that operates on a message byte at a time instead of bit by bit. The byte by byte method requires more memory by using a lookup table. The concept is that for a given input remainder and generator polynomial, the output remainder will always be the same. The CRC lookup table is computed once during system initialization and stored in memory. The CRC check value for a message is computed by passing each byte of the message to the lookup table to return a CRC value.

Advantages and Disadvantages of the CRC Method

The use of CRCs as a means of error detection is relatively fast and simple and takes little overhead in resources and time. The computation of a CRC is a little slower than the simple checksum where the words are simply summed or XORed. The shifting operation takes a little longer especially when implemented in software. The technique is stronger than a simple checksum or parity check in terms of error detection. The order of the message matters when computing a CRC, unlike a simple checksum or parity check. Computing a CRC starting from the end of the message will yield a different result than starting from the beginning of the

message. Similarly, a message in big endian format will have a different CRC than one in little endian format.

Hash Function

A cryptographic hash function can be used as another technique to detect errors in digital systems. A cryptographic hash function is an algorithm that takes a block of data and computes a fixed-length hash value. The block of data to be encoded is called the message and the computed hash value is called the message digest.

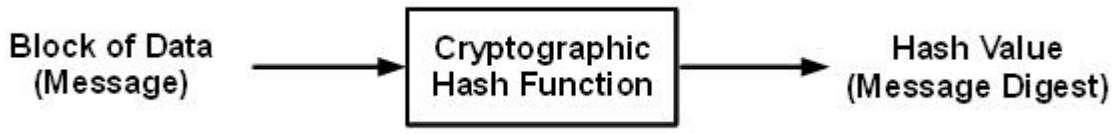


Figure 10 - Block Diagram of a Cryptographic Hash Function

The hash (or message digest) is a form of digital signature for the message that was sent through the hash function. Software programs may be "signed" using a hash function. During startup, the program may be sent through the hash function and the new signature is compared with one stored in a secure area on the device. The two signatures would be different if the program was tampered with or if a memory error occurred.

A hash algorithm must function such that it is extremely difficult to generate a message with a given hash. It must be extremely difficult to modify a message without changing the hash, and it must be unreasonable to find two different messages with the same hash.

Two popular hash functions are SHA-1 (secure hash algorithm) and MD5 (message digest algorithm). The message digest for SHA-1 is 20 bytes in length and that of the MD5 algorithm is 16 bytes. Both SHA-1 and MD5 are robust hash algorithms, but both have been found to contain security flaws or weaknesses.

Advantages and Disadvantages of Hash Functions

Cryptographic hash functions are many times more powerful in detecting errors, whether accidental or malicious, than a simple CRC check. The hash value is longer than a CRC check value. Typically a hash value is only computed once for the entire block of data, i.e., a software program or document. Therefore, if an error has occurred, the entire message must be discarded, not just a portion of it. Unlike a CRC over a small packet of data, hash functions are used to determine if an entire message (software program or document) is intact with no errors or tamper evidence. Hash functions usually take longer to execute than a CRC or checksum function.

Error Correction Techniques

There are two basic methods of controlling transmission errors in data transmission systems: the automatic repeat request (ARQ) method and the forward error correction (FEC) method.

Automatic Repeat Request Method

Using the ARQ method, if an error is detected for a block of data, the block of data is retransmitted. When a block of data is received, the data is checked using an error detection code. If an error is detected, the receiving end will send a request to the transmitting end to resend the data. This is usually accomplished via an ACK/NAK method.

An ACK (or acknowledge) is a small message with no data or payload that is sent back to the transmitter to signal the transmitter that the message was received with no errors. If an ACK is not received by the transmitter, the transmitter is required to resend the last message.

Similarly, a NAK (or NACK or no ACK or no acknowledge) is a small message with no payload that is sent back to the transmitter to signal the transmitter that the message was not properly received and needs to be resent. If no NAK is received by the transmitter, the transmitter may continue on to the next packet of data.

The ARQ method is sometimes called backward error correction (BEC).

Forward Error Correction Method

The FEC method of error correction employs the use of an error correcting code (ECC) which is built into the data transmission system. Data is encoded with an ECC and transmitted. If an error is detected on the receiving end the data may be "repaired" or recovered without retransmitting the original message. This is accomplished using simple mathematical techniques where certain bit sequences are not allowed. Single bit errors and multiple bit errors may be corrected with certain error correcting codes. Examples of ECCs include the Hamming codes, Golay codes and Reed-Solomon codes.

Often times in a communication system, a combination of automatic repeat request and forward error correction techniques will be employed.

Summary

Whether we notice it or not, data plays an important part of everyday life. Reading content from a website, downloading a book to an electronic reader, and texting a friend are all examples of data being transferred from one location to another. In order to ensure that the data was transferred to its destination error-free, simple error detection techniques may be utilized.

The use of a parity check is perhaps the simplest form of error detection. With this technique a bit is inserted after a fixed number of bits to maintain either an odd number or an even number of bits. A checksum in which all of the bytes in a given block of data are XORed (or summed) to produce a checksum value can be appended to a block of data before it is transmitted. When a block of data is received the checksum is recomputed and checked against the one just received with the block of data. Checksums are weak compared to a CRC. A CRC is used in the same

manner for error detection as a checksum, but are more robust because the value of the CRC depends on the order of the data.

Hash functions are also used for error detection, but typically for data at rest, not for data transfers. Hash functions are used for security-critical applications as a form of digital signature on software programs or documents to detect errors whether accidental or intentional.

References

1. "A Checksum Algorithm". 17 July 2011 <http://www.flounder.com/checksum.htm>

2. "A Checksum Algorithm". 27 March 2001 <http://www.codeproject.com/KB/recipes/checksum.aspx>

3. "An Illustrated Guide to Cryptographic Hashes". 9 May 2005 <http://unixwiz.net/techtips/iguide-crypto-hashes.html>

4. Campbell, Joe. *C Programmer's Guide to Serial Communications, Second Edition*. Indianapolis, IN: Sams Publishing, 1993.

5. "Checksum - Webopedia". visited 14 October 2011 <http://www.webopedia.com/TERM/C/checksum.html>

6. "Checksum - Wikipedia, the Free Encyclopedia". 8 September 2011 <http://en.wikipedia.org/wiki/Checksum>

7. "CRC Implementation Code in C". 2 December 2007 <http://www.netrino.com/embedded-systems/how-to/crc-calculation-c-code>

8. "Cryptographic Hash Function - Wikipedia, the Free Encyclopedia". 13 October 2011 <http://en.wikipedia.org/wiki/Cryptographic_hash_function>

9. "Cyclic Redundancy Check". 28 July 2009 <http://www.hackersdelight.org/crc.pdf>

10. "Cyclic Redundancy Check - Wikipedia, the Free Encyclopedia." 4 October 2011 <http://en.wikipedia.org/wiki/Cyclic_redundancy_check>

11. "Error Correcting Codes". 30 November 2007 <http://www.hackersdelight.org/ecc.pdf>

12. "Error Detection and Correction - Wikipedia, the Free Encyclopedia". 6 October 2011 <http://en.wikipedia.org/wiki/Error_detection_and_correction>

13. Lin, Shu and Costello, Daniel J. Jr. *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

14. "MD5". 15 October 2011 <http://en.wikipedia.org/wiki/MD5>

15. "Parity Bit - Wikipedia, the Free Encyclopedia". 30 September 2011 <http://en.wikipedia.org/wiki/Parity_bit>

16. "SHA-1". 7 October 2011 <http://en.wikipedia.org/wiki/SHA-1>

Final Exam – Detection of Errors in Digital Systems

1. In the development of the information age, Claude Shannon introduced the concept of _____ to information which is equivalent to a shortage of information content in a message, thus introducing errors to the message.
    a. redundancy
    b. enthalpy
    c. entropy
    d. compression

2. _____ is not an error detection technique.
    a. Parity check
    b. Cyclic redundancy check
    c. ACK / NAK
    d. Checksum

3. Compute the checksum for the following data block (using the running XOR method):
AE 6F 5A 80 7F 5B 55 4C.
    a. 7A
    b. AE
    c. 52
    d. 26

4. Compared with a checksum and a CRC, a parity check is the fastest, easiest form of error detection, but is the least reliable.
    a. True
    b. False

5. Of the following _____ is the most robust error detection technique.
    a. checksum
    b. CRC
    c. parity check
    d. ACK / NAK

6. Anomalies such as _____ can be detected in a data transfer by using a checksum.
    a. data transposed from big endian to little endian
    b. single and multiple bit errors
    c. data bytes out of order in the message
    d. all of the above

7. The heart of the CRC algorithm is the _____.
    a. generator polynomial
    b. microprocessor
    c. hash function
    d. message digest

8. A parity bit is computed by performing a logical _____ on a set of bits.
    a. XOR
    b. AND
    c. OR
    d. NAND

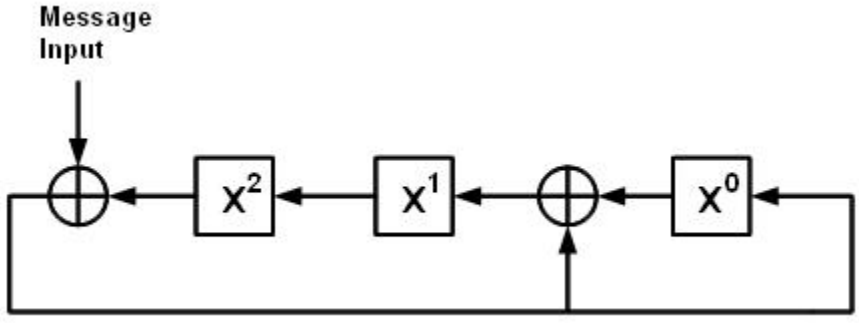
9. The use of a CRC in a data transfer can detect anomalies such as _____.
    a. data transposed from big endian to little endian
    b. single and multiple bit errors
    c. data bytes out of order in the message
    d. all of the above

10. In order to maintain even parity on the following set of bits: 0 1 0 1 1 0, the parity bit would be set to _____.
    a. 1
    b. 0

11. Error detection techniques are used to _____.
    a. reliably recover data stored in a memory device
    b. ensure the reliable delivery of data over a communications channel
    c. add overhead
    d. both A & B

12. The automatic repeat request (ARQ) is a technique used for error correction in which _____.
    a. any errors that are detected on the receiving end are corrected without retransmission
    b. an error correction code (ECC) is used to correct any errors that are received
    c. no extra data is transmitted, only the message
    d. the receiving end will check a block of data using an error detection code and when an error is detected the receiving end will request for the data to be resent

13. _____is the generator polynomial for the following CRC circuit:



    a. $G = x^3 + x^2 + 1$
    b. $G = x^8 + x^3 + x + 1$

c. $G = x^3 + x + 1$
d. $G = x^2 + x + 1$

14. Using the CRC circuit shown above, with an initial value of [0 0 0] and a message of [1 0 1 0], the final CRC value will be _____.
    a. 1 0 1
    b. 0 1 1
    c. 1 1 1
    d. 1 0 0

15. A cryptographic hash function has all of the following properties except _____.
    a. it is extremely difficult to modify a message without changing the hash
    b. the hash function computes a variable-length hash value
    c. it is extremely difficult to generate a message with a given hash
    d. it is extremely difficult to find two different messages with the same hash