2. Public-key Cryptosystems - Hash functions

2.1 OVERVIEW

2.1.1 Introduction and learning objective

Symmetric encryption has evolved from classical to modern but also leaves a lot to be desire. Two of the most difficult problems associated with symmetric encryption are:

- ***Key distribution*** *- the issue of secret key exchange between sender and receiver*: Key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a key distribution center. A secure channel is required for key exchange so that the key must be kept secret and known only to the sender and receiver. This will be difficult to implement and establishing such a secure channel will be costly and time-consuming.
- ***Digital signatures*** *and confidentiality of keys and* : If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents. Besides, there is no basis to blame if the key is revealed.

Diffie and Hellman achieved an astounding breakthrough in 1976, by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography, going back over four millennia. That is *public key cryptography* or *asymmetric cryptography*. The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography.

To discriminate between the two, we refer to the key used in symmetric encryption as a **secret key**. The two keys used for asymmetric encryption are referred to as the **public key** (are often denoted as *PU*) and the **private key** (are often denoted as *PR*). The plaintext is denoted by M, the ciphertext is denoted by C

There are 2 main approaches in Public-key cryptography:

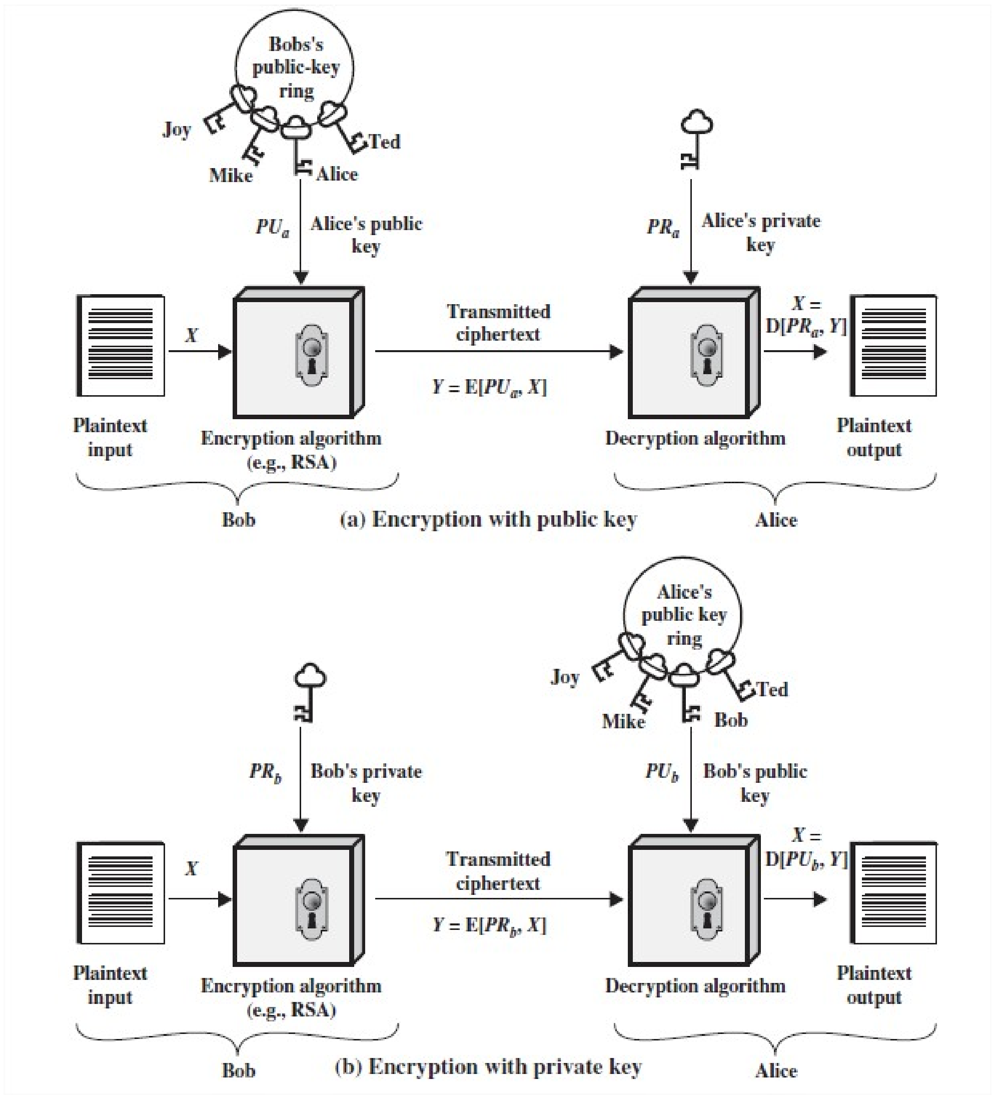


Figure 2.1: Two approaches in Public-Key Cryptography

1. **Public-Key Cryptosystem for Confidentiality**:
   Encryption with public key (Figure 2.1). If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.
   - Encryption: **C = E(M,** *PU*)
   - Decryption: **M = D(C,** *PR*)
2. **Public-Key Cryptosystem for Authentication**:
   Encryption with public key (Figure 2.1). In this case, Alice prepares a message to Bob and encrypts it using Alice's private key before transmitting it. Bob can decrypt the message using Alice's public key. Because the message was encrypted using Alice's private key, only Alice could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature**.

- Encryption: **C = E(M,** *PR*)
- Decryption: **M = D(C,** *PU*)

The *learning objective* of this lab is for students to gain hands-on experiences on the RSA algorithm, Diffie-Hellman key exchange and Hash functions. From lectures, students should have learned the theoretic part of the RSA algorithm, so they know mathematically how to generate public/private keys and how to perform encryption/decryption and signature generation/verification. This lab enhances student's understanding of RSA by requiring them to go through every essential step of the RSA algorithm on actual numbers, so they can apply the theories learned from the class. Besides, students should have learned the theoretic part of the DH algorithm, so they know mathematically how Alice and Bob share the secret key through an insecure channel. Additionally, other goal is for student to really understand the impact of collision attacks, and see in first hand what damages can be caused if a widely-used one-way hash function's collision-resistance property is broken (MD5 and SHA-1 collision).

2.1.2 Background

To achieve better results in this lab, you are expected to gain knowledge about:
- RSA Cipher
- The Diffie-Hellman key exchange
- Cryptographic Hash Functions

If you are not familiar with them, you should find out in more detail in:
Chapter 9: Public-Key Cryptography and RSA;
Chapter 10: Other Public-key Cryptosystems;
Chapter 11: Cryptographic Hash Functions
W. Stallings, *CS book - Cryptography and network security: Principles and practice, 7th ed*. Boston, MA, United States: Prentice Hall, 2017

2.1.3 Lab environment and Tools

1. **Operating system:** Windows, Linux (Ubuntu), MacOS
2. **Programing languages and IDE:** Flexible, you are free to choose any programming language you wish (Python, Golang are highly recommended)
3. **Tools:** CrypTool 2 - `https://www.cryptool.org/`.

2.2 LAB TASKS

2.2.1 RSA Public-Key Encryption

RSA is one of the first public-key cryptosystems and is widely used for secure communication. This cipher was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. In RSA scheme, the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}. RSA algorithm can be summarized as follows.

1. Select two "large" prime numbers p and q ($p \neq q$), then calculate n = pq
2. Calculate $\phi(n) = (p-1)(q-1)$
3. Select e such that e is relatively prime to $\phi(n)$ and less than $\phi(n)$.
4. Determine d such that $e.d \equiv 1 \, mod \, \phi(n)$ *(d can be calculated using the extended Euclid's algorithm)*
5. The resulting keys are public key $PU = (e,n)$ and private key $PR = (d,n)$

6. To encrypt a plaintext input of M:
   + Encryption for Confidentiality: $C = E(M,PU) = M^e mod\, n$
   + Encryption for Authentication: $C = E(M,PR) = M^d mod\, n$
7. To decrypt ciphertext input of C:
   + Decryption for Confidentiality: $M = D(C,PR) = C^d mod\, n$
   + Decryption for Authentication: $M = D(C,PU) = C^e mod\, n$

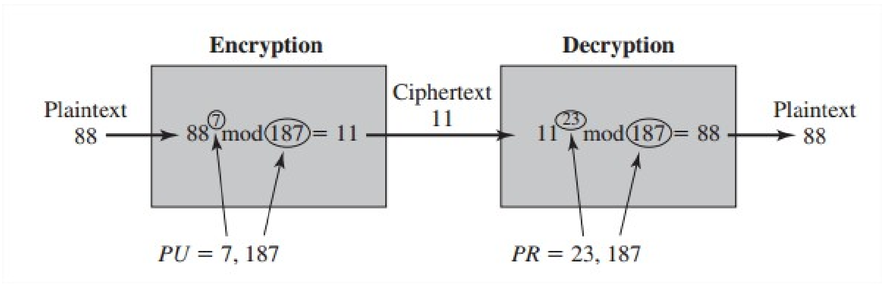


Figure 2.2: Example of RSA Algorithm

Task 2.1 To get acquainted with RSA, your task is using CrypTool 2 or other tools to perform the following experiments:

1. Determine public key PU and private key PR, if:
   + $p_1 = 11, q_1 = 17, e_1 = 7$ (decimal)
   + $p_2 = 20079993872842322116151219$,
   $q_2 = 676717145751736242170789, e_2 = 17$ (decimal)
   + $p_3 = F7E75FDC469067FFDC4E847C51F452DF$,
   $q_3 = E85CED54AF57E53E092113E62F436F4F, e_3 = 0D88C3$ (hexadecimal)
   (Note: Remember to check if the above values are prime numbers or not before calculating the keys)
2. Using key which generated by p_1, q_1, e_1 to encrypt and decrypt the plaintext M=5 in both case *Encryption for Confidentiality* and *Encryption for Authentication*.
3. Use the keys above to encrypt the following message:
   The University of Information Technology
   Determine the ciphertext as Base64.
4. Find the corresponding plain-text of each following ciphertext, knowing that they are encrypted by one of the three given keys above.
   (a) raUcesUlOkx/8ZhgodMoo0Uu18sC20yXlQFevSu7W/
   FDxIy0YRHMyXcHdD9PBvIT2aUft5fCQEGomiVVPv4I
   (b) C8 7F 57 0F C4 F6 99 CE C2 40 20 C6 F5 42 21 AB AB 2C E0 C3
   (c) Z2BUSkJcg0w4XEpgm0JcMExEQmBlVH6dYEp
   NTHpMHptMQ7NgTHlgQrNMQ2BKTQ==
   (d) 00101000 00010100 11111111 10110111 00101110 11001010
   11101100 01100111 10111111 00111111 01101000 11001111 00110000
   10010100 01010100 11110101 01001100 11101110 11101111 01011011
   00000100

Tips 2.1. *You can use the RSA Cipher template or build their own template to solve these problem in CrypTool 2*

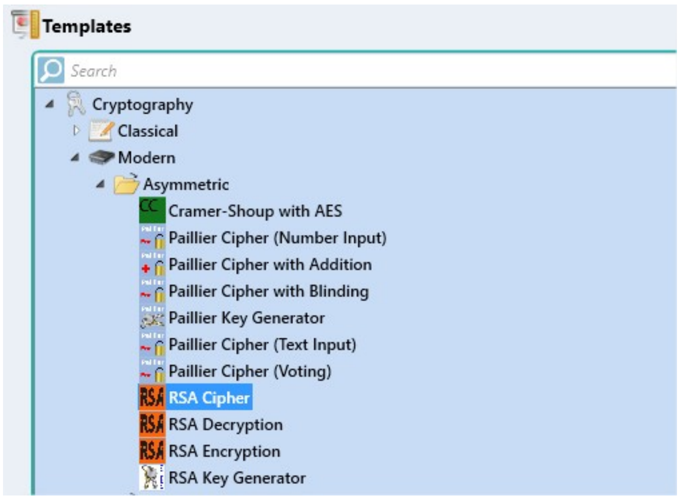


Figure 2.3: RSA Cipher template in CrypTool 2

2.2.2 Diffie-Hellman Key Exchange

The Diffie-Hellman (DH) key exchange is a method of securely exchanging cryptographic keys (can be used for symmetric ciphers) over a public channel and was one of the earliest and simplest Public Key Cryptography Standards (PKCS). DH algorithm depends for its effectiveness on the computing **discrete logarithms**.

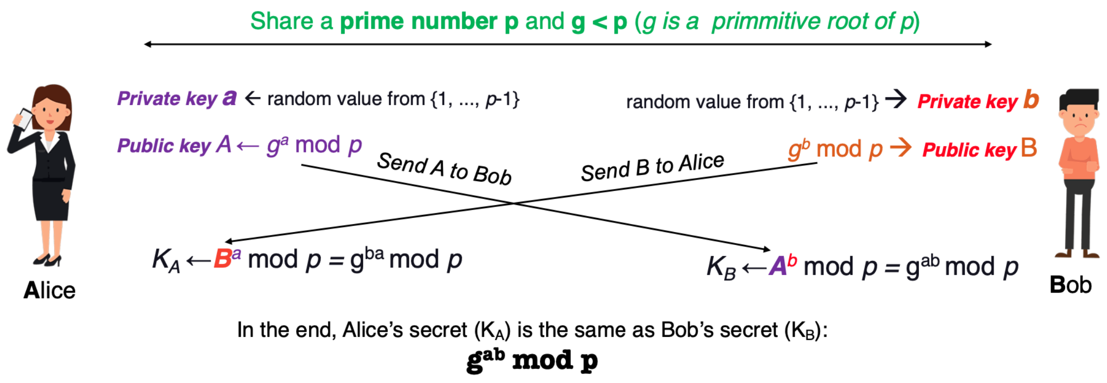

Share a prime number p and g < p (g is a primmitive root of p)
Private key a ← random value from {1, ..., p-1}
random value from {1, ..., p-1} → Private key b
Public key A ← g^a mod p
g^b mod p → Public key B
Send A to Bob
Send B to Alice
K_A ← B^a mod p = g^ba mod p
K_B ← A^b mod p = g^ab mod p
Alice
Bob
In the end, Alice's secret (K_A) is the same as Bob's secret (K_B):
g^ab mod p

Figure 2.4: The Diffie–Hellman Key Exchange

The DH key exchange can be briefly described as follows:

1. Firstly, Alice and Bob share a prime number **p** and an integer **g** such that **g<p** and **g** is a **primitive root** of **p**.
   Note that ***p*** *and* ***g*** *are publicly known and don't need to be kept secret.*
2. Subsequently, Alice selects a random integer **a** such that **a < p** and keep it secret. Bob also independently selects a random integer **b** such that **b < p** and keep it secret (**a** and **b** are called private keys)
3. Then, Alice computes $A = g^a mod p$ and send to Bob. Similarly, Bob computes a $B = g^b mod p$ and send to Alice. Thus a and A is Alice's corresponding public key, and similarly for Bob.

4. Finally, each side compute the shared secret K. Alice computes:
   $K_A = B^a mod\, p = (g^b)^a mod\, p = g^{ba} mod\, p$
   Bob computes:
   $K_B = A^b mod\, p = (g^a)^b mod\, p = g^{ab} mod\, p$
   These two calculations produce identical results: $K = g^{ab} mod\, p$. The result is that the two sides have exchanged a secret value. Typically, this secret value is used as shared symmetric secret key.

Task 2.2 To get familiar with Diffie-Hellman key exchange, your task is determine the shared key of Alice and Bob (K) and their public keys (A,B) with the following information:

- p = 1663, g = 131, a = 12, b = 19
- p = 9377, g = 3511, a = 20, b = 2
- p = 77818714124031252720819944202374984988485179601906368139945936248931661607804455138878214123130093616259366874525861746969093031416911703465687883907 64347,
  g = 11124136727432308918812736531716408346878332998901256519320663274861933580804712151007972377757714671498942601235126601364021584069983694812528444527 18796,
  a = 38, b = 87

In Diffie-Hellman (DH) key exchange, if an adversary knows the following ingredients $g, p, g^a mod\, p, g^b mod\, p$, then will he able to determine the secret key $g^{ab} mod\, p$? Why? Is DH totally secure? Find out and describe the attacks against DH *(if any)*. ■

Tips 2.2. *You can use the Diffie Hellman Key-Exchange template or build their own template to solve these problem in CrypTool 2.*
The following diagram shows an example of the attack against Diffie-Hellman key exchange protocol:

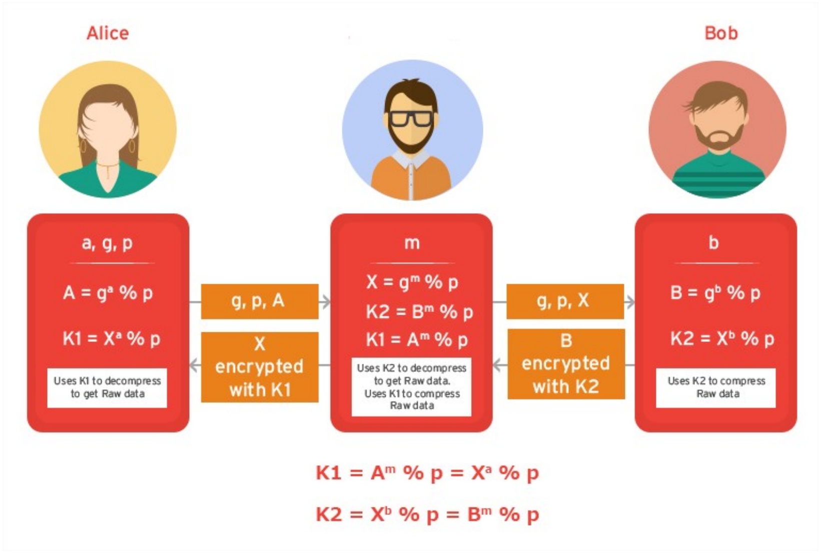


Figure 2.5: An attack against Diffie-Hellman key exchange

2.2.3 Hash functions

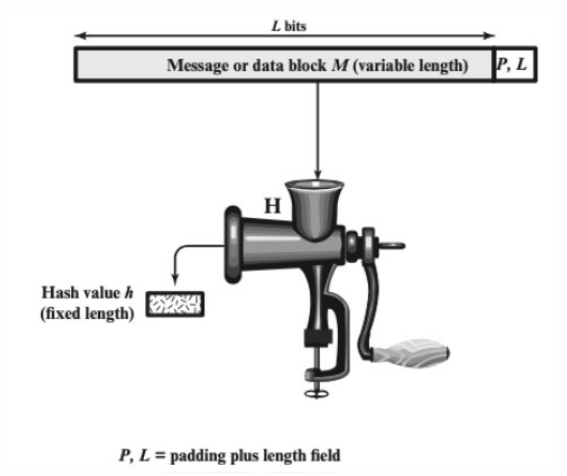

L bits
Message or data block M (variable length)
P, L
H
Hash value h
(fixed length)
P, L = padding plus length field

Figure 2.6: Hash Function

For cryptographic applications, need one or more of these properties:

1. **The one-way property** : Given **h**, it's infeasible to find **x** such that **H(x)=h**. *(Also called the "Preimage resistance")*
2. **The collision-free property**:
   - *Weak collision resistance*: Given x, it's infeasible to find $y \neq x$ such that $H(x) = H(y)$. *(Also called "Second preimage resistance")*
   - *Strong collision resistance* : It's infeasible to find any two x and y such that $x \neq y$ and $H(x) = H(y)$. *(Also called "Collision resistance")*

R A secure one-way hash function needs to satisfy two properties: the one-way property and the collision-resistance property. The one-way property ensures that given a hash value h, it is computationally infeasible to find an input M, such that $hash(M) = h$. The collision-resistance property ensures that it is computationally infeasible to find two different inputs M1 and M2, such that $hash(M1) = hash(M2)$.

Several widely-used one-way hash functions have trouble maintaining the collision-resistance property. At the rump session of CRYPTO 2004, Xiaoyun Wang and co-authors demonstrated a collision attack against MD5 [3]. In February 2017, CWI Amsterdam and Google Research announced the SHAttered attack, which breaks the collision-resistance property of SHA-1 [4]. While many students do not have trouble understanding the importance of the one-way property, they cannot easily grasp why the collision-resistance property is necessary, and what impact these attacks can cause. *(SEED Labs - Wenliang Du, Syracuse University)*

Task 2.3 Generating message digests (hash values)

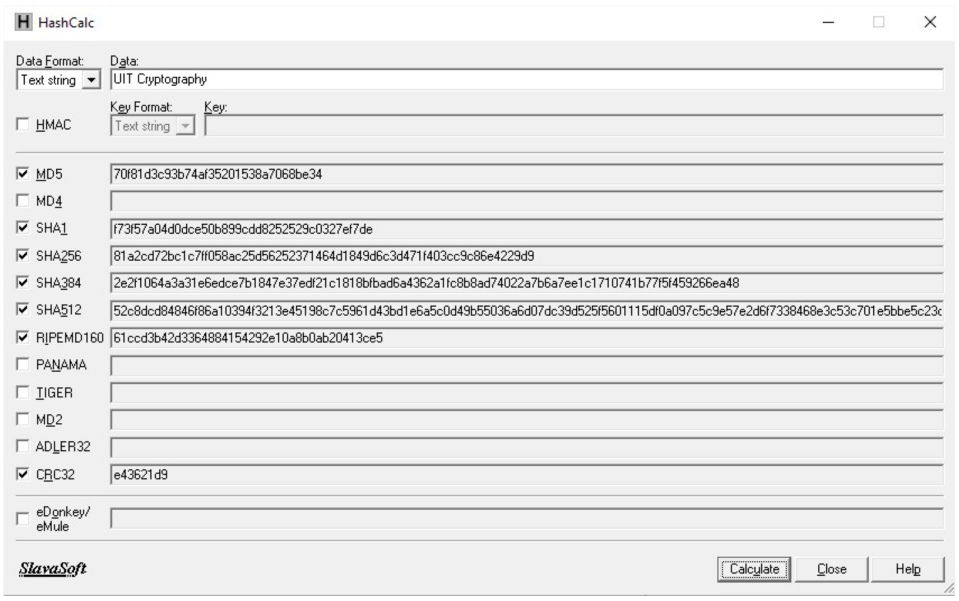
1. Your task is calculating hash values (MD5, SHA-1, SHA-256, SHA384, SHA512) for each of the following inputs:
   - Text string: <Your Name>. For example, "Nguyen Van A"
   - Hex string: Your Student ID (assumed it is in HEX format). For example, "19 52 12 34"

   Then, change an arbitrary character in each input and calculates all hash values again. Compare the results with the original hashes and describe your observations.

2. Create a text file, put your name and student's ID inside. For example, "Nguyen Van An - 19521234". Generate hash values H1 of this files. Subsequently, send this file to your friend via email or Facebook or upload to Google Drive and then download it back. Calculate hash values of the downloaded file and compare to those of the original file.
   Please observe whether these hash values are similar or not.

You are able to use Cryptool 2 or similar tools like HashCalc in this task. ■

Tips 2.3. *You can refer to a similar application like HashCalc (`https://www.slavasoft.com/hashcalc/`)*



```
Data Format: Text string    Data: UIT Cryptography
[ ] HMAC    Key Format: Text string    Key:

[✓] MD5         70f81d3c93b74af35201538a7068be34
[ ] MD4
[✓] SHA1        f73f57a04d0dce50b899cdd8252529c0327ef7de
[✓] SHA256      81a2cd72bc1c7ff058ac25d56252371464d1849d6c3d471f403cc9c86e4229d9
[✓] SHA384      2e2f1064a3a31e6edce7b1847e37edf21c1818bfbad6a4362a1fc8b8ad74022a7b6a7ee1c1710741b77f5f459266ea48
[✓] SHA512      52c8dcd84846f86a10394f3213e45198c7c5961d43bd1e6a5c0d49b55036a6d07dc39d525f5601115df0a097c5c9e57e2d6f7338468e3c53c701e5bbe5c23c
[✓] RIPEMD160   61ccd3b42d3364884154292e10a8b0ab20413ce5
[ ] PANAMA
[ ] TIGER
[ ] MD2
[ ] ADLER32
[✓] CRC32       e43621d9

[ ] eDonkey/
    eMule
```

Figure 2.7: HashCalc application on Windows OS

Hash properties: One-way vs Collision-free

Task 2.4 It is now well-known that the crytographic hash function MD5 and SHA-1 has been clearly broken (in terms of collision-resistance property). We will find out about MD5 and SHA-1 collision in this task by doing the following exercises:

1. Consider two HEX messages as follow:
   Message 1
   d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
   55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
   d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
   e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
   Message 2
   d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
   55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
   d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
   e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70

How many bytes are the different between two messages?
Let's generate MD5 hash values for each message. Please observe whether these MD5 are similar or not and describe your observations in the lab report.

2. Consider two executable programs named hello and erase.
   - If you are using Windows, you can download these .exe files here here.
   - If you are using Linux, you can download the similar pair: **hello** and **erase**.

   Run these programs and observe what happens. Note these programs must be run from the console. Let's generate MD5 hash values for these programs and report your observations
3. Download two PDF files: shattered-1.pdf and shattered-2.pdf. Open these files to check the different. Then generate SHA-1 hash for them, observe the result.

Draw the conclusion base on your observations. Could you explain the reasons for the existence of collision in MD5 and SHA-1? ■

2.3 REQUIREMENTS - EVALUATION

2.3.1 Requirements

You are expected to complete all tasks in section 1.2, advanced tasks are optional and you could get bonus points for completing those tasks. You can either practice individually or work in team (2 members/team). If you prefer to work in team, please register in the first class with instructor and keep working in your team afterwards.

Your submission must meet the following requirements:

- You need to submit a **detailed lab report in .PDF** format, **using report template** that was provided on the courses website. You need to provide screenshots, to describe what you have done and what you have observed and explanation to the observations that are interesting or surprising.
- **Both of Vietnamese and English report are accepted**, that's up to you. Students in BCU, High Quality and Honor program are expected to write lab report in English.
- When it comes to **programming tasks** *(require you to write an application)*, please attach all source-code and executable files (if any) in your submission. Please also list the important code snippets followed by explanation and screenshots when running your application. Simply attaching code without any explanation will not receive points.
- **Submit work you are proud of - don't be sloppy and lazy!**
  Your submissions must be your own work. You are free to discuss with other classmates to find the solution. However, report-copy is prohibited. Both of report owner and copier are received *a special gift - Zero point*. Please remember to clearly cite any source of material (website, book,...) that influences your solution.

Notice 2.3.1 Combine your lab report and all related files into a single ZIP file (.zip), name it as follow and submit to courses website (`https://courses.uit.edu.vn`):

StudentID1_StudentID2_ReportLabX

For example: 19520123_19520234_ReportLab1.zip. Maximum file size: 10MB.
If it is larger than 10 MB, then you should upload to Google Drive and submit the link to your report *(remember to share view permission with instructor)*

2.3.2 Evaluation

- Well complete all basic tasks: 80%
- Well complete the advanced tasks: 10-50% or bonus points to the next lab.
- In-class activities: + up to 100 %
- Report written in English: + up to 20%

Notice 2.3.2 Assignments are expected to be completed by due date. For every day the assignment is late after the deadline, 10% will be deducted from the lab score. No assignments will be accepted once they are 7 or more days late.

Any part presented in your report may be randomly examine at the next class to verify your work. Absence without any rational reason could result in 30% deduction (or more) of your team's score.

2.4 REFERENCES

[1] William Stallings, *Cryptography and network security: Principles and practice, 7th ed*, Pearson Education, 2017. *Chapter 9: Public-key Cryptography and RSA, Chapter 10: Other Public-key Cryptosystems, Chapter 11: Cryptographic Hash Functions*
[2] Wenliang Du (Syracuse University), *SEED Cryptography Labs*
`https://seedsecuritylabs.org/Labs_16.04/Crypto/`.
[3] Bernhard Esslinger et al., *The CrypTool Book: Learning and Experiencing Cryptography with CrypTool and SageMath, 12th ed, 2018*. Available: `https://www.cryptool.org/en/ctp-documentation`.

Training platforms and related materials

- ASecuritySite - `https://asecuritysite.com`
- Cryptopals - `https://cryptopals.com`

Attention: *Don't share any materials (slides, readings, assignments, labs,...) out of our class without my permission!*