**Unit 1: Cipher Fundamentals**

**The key concepts are:**

* + Ciphers.
  + Encoding methods (ASCII, UTF-16, Base64, Hex).
  + Prime Numbers.
  + GCD.
  + Large numbers.
  + Random Number Generators.
  + Data Integrity (CRC-32).
  + Frequency Analysis.
  + Key-based encryption.
  + Key sizes.

**What you should know at the end of unit?**

* + Understand the conversion of characters between hex, decimal and octal. Sample question: Convert "hello" into a hex stream. Related material: [here](https://asecuritysite.com/Coding/ascii).
    - *Why are we studying this?* Encrypted content is converted into a range of different formats, so we need to understand the process of taking plain text and then converting it into other encoding formats. Encryption keys, ciphertext and digital certificates are examples of binary content which must be represented in a text format.
  + Compute the GCD for values. Sample question: What is the GCD for 42 and 56? Related material: [here](https://asecuritysite.com/encryption/gcd).
    - *Why are we studying this?* GCD is a fundamental building block used in public key encryption, where we must find two numbers who do not share a common divisor. When we look at public key encryption we will see how GCD is used.
  + Compute the MOD for values. Sample question: What is the result of 13 MOD 7?
    - *Why are we studying this?* Within many of the public key methods we use the MOD operator with a prime number, and where it is difficult to find the value of x for Y=gx(mod p), even though we know Y, g and p.
  + Understand how to manually convert from ASCII to Base-64, and vice-versa. Sample question: What is the Base-64 conversion of “hello”? [here](https://asecuritysite.com/Coding/ascii).
    - *Why are we studying this?* Base-64 is used extensively in encryption, and many of the keys and cipher text are transported and stored in a Base-64 format.
  + Calculate the time taken to crack a code given a time to try each key, and for the number of processing elements. Sample question: If it takes 100 years to crack a cipher code, and computing power doubles each year. How long will it take to crack a code after five years?
    - *Why are we studying this?* We always need to understand the strengths of your encrypted data, especially in the face of GPU based crackers, so we need to understand how quickly it will take to crack our cipher.

Presentations

* + Week 1 Presentation (PDF): [here](https://github.com/billbuchanan/esecurity/tree/master/unit01_cipher_fundamentals/lecture)
  + Week 1 Presentation (video): [here](https://www.youtube.com/watch?v=zqmjUpJNcJA)
  + Week 1 Class Lecture (video): [here](https://youtu.be/3hkRjzl8B8w)

Lab

* + Week 1 Lab (PDF): [here](https://github.com/billbuchanan/esecurity/tree/master/unit01_cipher_fundamentals/lab) [demo](https://www.youtube.com/watch?v=v6H7lHblKes)

**Sample exam questions**

* + Using the table [here](https://asecuritysite.com/public/test_table.pdf), what is the Base-64 encoding for "test"?

>>> from base64 import b64encode  
>>> b64encode('test')  
'dGVzdA=='

>>> from base64 import b64decode  
>>> b64decode('dGVzdA==')  
'test'

* + Using the table [here](https://asecuritysite.com/public/test_table.pdf), is the Base-64 encoding for "help"?

def base64encode(message):

b64char=("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+-")

msg = message

bits = ""

for c in msg:

bits+="{0:b}".format(ord(c)).zfill(8) #zero pad char to 8 bits

encoded = ""

while len(bits) > 0:

chBits=bits[0:6]

bits=bits[6:]

if len(chBits)<6:

chBits.ljust(6,"0") #zero pad

encoded += b64char[int(chBits,2)]

encoded += "=="[0:4-(len(encoded)%4)]

return encoded

* + If it takes 1ns to test an encryption key. How long will it take to crack a 32-bit key?

>>> k=2\*\*32

>>> k

4294967296

>>> sec=k/1000

>>> min=sec/60

>>> hr = min/60

>>> day = hr/24

>>> day

49

>>> avg = day/2

>>> avg

24 days

* + If it takes 10ns to test an encryption key. How long will it take to crack a 20-bit key?

52 seconds

* + Bob tells Alice that she won't be able to view the cipher text, but when she looks at the messages, they seem to be full of printable characters. What format is Bob likely to be using for the encoding of the cipher text, and what would you ask Alice to look for, in order to confirm your guess?

He’s most likely to be using Base 64 as an encoding method. One or two “=” signs at the end of the cipher text are a good indicator, but there may be none. Also, she should look for upper and lower case characters, and “+” / “-“

* + Alice has been reading her crypto books, and she reads that there should be an '=' symbol at the end of the encoding. She observes her encoding of cipher messages to Bob, and sees that some do not have an '=' sign at the end. Is there a problem with her encoder? If not, how often, on average, should she see an '=' sign at the end of her ciphered messages?

They’re used to bad blocks of characters to 4, and end in 0, 1 or 2 “=”. She should one or two approximately 66% of the time.

* + Bob has two numbers which give a GCD of 1. Trent says that this happens because the numbers are prime. Is Trent correct? Explain your answer.

The numbers are relatively prime (ie. They don’t share a common factor other than 1), however they may not be prime numbers

* + Bob deals in Bitcoins and tells Alice that he has a Base-58 ID? Alice says he is crazy, and has only heard of Base-64. What is Base-58 and how does it differ from Base-64?

Base 58 is similar to Base 64, but has some characters that look similar to others removed (ie. The letter “O” and the number zero, the lowercase ‘L’ and number 1)

* + Bob encrypted a message in 1980, and it took a million years to crack at the time. Assuming that computing power doubles each year, do you think the message will be safe against cracking for existing computer systems?

No, by 2009, it would be cracked in under than a day. Also, this 1M year crack time may well have been based on a single processor system, so with the advent of cloud computing and mutlti GPU matrix-processors available for hire, much stronger encryption methods are crackable now.

*From <*[*https://github.com/billbuchanan/esecurity/tree/master/unit01\_cipher\_fundamentals*](https://github.com/billbuchanan/esecurity/tree/master/unit01_cipher_fundamentals)*>*

**Unit 2: Symmetric Key**

The key concepts involved are defining key entropy; key generators (such as using hashing methods to generate keys based on passphrases); symmetric key methods (AES, Twofish, 3DES, RC4 and ChaCha20); stream or block encryption; symmetric key modes (ECB/CBC/OFB); and salting/IV.

**What you should know at the end of unit?**

* + The differences between a stream cipher and a block cipher.
  + How salting is used to change the cipher blocks.
  + Use openssl to perform practical operations.
  + Understand the encoding formats used for cipher text and keys.
  + Define the difference between cipher block modes, such as between ECB and CBC.

Presentations

* + Week 2 Presentation (PPTX) - Symmetric Key Encryption: [here](https://github.com/billbuchanan/esecurity/blob/master/unit02_symmetric/lecture/chapter02_secret.pptx)
  + Week 2 Presentation (Video) - Symmetric Key Encryption [here](https://youtu.be/nLRV34K3xIo)
  + Week 2 Presentation (Lecture - Video) - Symmetric Key Encryption: [here](https://youtu.be/CCOt8Xk3ZVU)

Lab

* + Unit 2 Lab (PDF): [here](https://github.com/billbuchanan/esecurity/blob/master/unit02_symmetric/lab/new_lab02.pdf)
  + Unit 2 Lab (Video): [here](https://youtu.be/N3UADaXmOik)

Quick demos

* + Introduction to AES: [here](https://www.youtube.com/watch?v=rSyvUYbMok8)
  + Padding in ciphers: [here](https://www.youtube.com/watch?v=R3NosHMSi0o)
  + Why EDE in 3DES?: [here](https://www.youtube.com/watch?v=ttayDxqfQkA)

**Sample exam questions**

The following are sample exam questions for symmetric key:

* + Explain the differences between stream and block ciphers, and why salt is required within the encryption process.
    - Where would I find this info? Have a look at the penguin in [Unit 2](https://asecuritysite.com/public/chapter02_secret.pdf) (Slide 31), and here's an outline of the problem with ECB in this related [article](https://medium.com/asecuritysite-when-bob-met-alice/when-is-high-grade-encryption-not-high-grade-when-its-ecb-e1509ec56930?source=friends_link&sk=31ec28f1c2be74a81e53c67e71d5b259).
  + What are the possible advantages of using stream ciphers over block ciphers?
  + Bob encrypts his data using secret key encryption and sends it to Alice. Every time he produces the cipher text it changes, and he is worried that Alice will not be able to decipher the cipher text. He encrypts "Hello" and gets a different cipher stream each time. Why does the cipher text change, and why is she still able to decrypt it, even though it changes each time?
  + AES uses an S-box to scramble the bits. How are the S-boxes for the encryption and decryption process linked?
  + Bob is sending encrypted data to Alice, and Eve is listening. After listening for a while, Eve is able to send a valid encrypted message to Alice. By outlining ECB, discuss how this might be possible.
    - Where would I find this info? Have a look at the penguin in [Unit 2](https://asecuritysite.com/public/chapter02_secret.pdf) (Slide 31), and here's an outline of the problem with ECB in this related article.
  + Bob is using a password to generate a 128-bit encryption key. Explain why the key space is unlikely to be 2128, and why key entropy could be used to measure the equivalent key size.
    - Where would I find this info? This is related to key entropy [here](https://asecuritysite.com/encryption/en), and try and understand how key entropy relates to the strength of the encryption.
  + Bob says that the number of bytes used for the cipher text will change directly with the number of bytes used in the plain text. Alice disagrees and says that most encryption methods involve having block sizes. Who is correct? Explain why.
  + With block encryption, how do we know where the ciphered data actually ends? Does it just use an end-of-file character or a NULL character?
  + Alice says she is confused that Bob is sending her the same message as a cipher, but every time the cipher text changes. Apart from using the shared encryption key, what does Alice use to decipher the cipher text?
  + Bob tells Alice that she won't be able to view the cipher text, but when she looks at the messages, they seem to be full of printable characters. What format is Bob likely to be using for the encoding of the cipher text, and what would you ask Alice to look for, in order to confirm your guess?
  + Which of these is correct for CMS padding: "68656c6c6f3132330808080808080808", "68656c6c6f3132330909090909090909", and "68656c6c6f3132330A0A0A0A0A0A0A0A".
    - Where would I find information on this? Look [here](https://asecuritysite.com/encryption/padding).
  + Bob wants to cipher "edinburgh" with the key of "hello123" for a 256-bit AES key, and his encoding gives him "6564696e6275726768". What will be the padding that will be added?
  + Eve says she thinks she can determine the number of characters within some ciphered plain-text. Is she correct? If so, how many plain-text characters were there in this ciphered message: "6920776f756c64206c696b6520746f2074616b65206120627265616b04040404".
  + RC4 is a stream cipher, which is one of the recommended ciphers for IoT devices. Bob says that it has an infinitely long encryption key, and that his devices will not be able to cope with this size of key. How would you convince him that IoT devices will be able to cope with RC4?
  + RC4 is used within Wifi systems. With WEP, a 40-bit encryption key which was shared over the network, and which had a 24-bit IV value. In relation to the key size, the scope of the key, and the size of the IV, what do you think were the fundamental problems with this setup?
  + Bob says that he can creat two ciphers from a file with the word "hello", and which will always create the same cipher. If the cipher is "Z8onq9tXC3CL2oOwqLLWbg==" and the key is "password", which is the missing part of the command he used (find the replacement for [OPTION1] and [OPTION2]):

openssl enc -e -[OPTION1] -in test.txt -pass pass:password -nosalt -[OPTION2]

The following are encrypted with aes-256-cbc or 3-DES and have a password of "napier", "123456" or "password". Decode them:

* + U2FsdGVkX18K9Dy9I/CewpNH2svvjyhNG3Bod77+uYo=
  + U2FsdGVkX18pmUpnI7iopG3gsHVQPT1zyRwjlvAJ+aI=
  + U2FsdGVkX19XlsCN50CFxZlBcCplPs9/

Please note: In the file you create, put one new line after the Base64 text. For example the answer to the first one is:

openssl enc -d -aes-256-cbc -in test.txt -pass pass:123456 -base64

*From <*[*https://github.com/billbuchanan/esecurity/tree/master/unit02\_symmetric*](https://github.com/billbuchanan/esecurity/tree/master/unit02_symmetric)*>*

Addendum

In the lecture, the slide at the end of Unit 2 (Symmetric Key) should be (for an eight character password and with [a-z]):

Tests

* + Test (Symmetric Key Encryption): [here](https://asecuritysite.com/tests/tests?sortBy=cryptobook02)

Note: There will be no multiple choice questions in the tests.

*From <*[*https://github.com/billbuchanan/esecurity/tree/master/unit02\_symmetric*](https://github.com/billbuchanan/esecurity/tree/master/unit02_symmetric)*>*

# 

**Unit 3: Hashing and MAC**

The key concepts are: MD2. MD4. MD5. SHA-1. Salting. Collisions. Murmur and FNV. Bloom Filter. LM Hash. SHA-3. Bcrypt. PBKDF2. Open SSL Hash passwords. One Time Passwords. Timed One Time Password (TOTP). Hashed One Time Password (HOTP). HMAC.

**What you should know at the end of unit?**

* How the lengths of the hashes vary with the number of bits in the hash.
* How we can calculate the strengths on passwords.
* Understand how salt is applied to the hashing process.
* Define how collisions can occur within hashing.
* Implement hash cracking methods (John the Ripper and Hashcat).
* Defines the usage of signed hashes (eg HMAC).
* Outlines the usage of OTP and Timed Passwords.

**Presentations**

* Week 3 Presentation (PDF) - Hashing: [here](https://github.com/billbuchanan/esecurity/blob/master/unit03_hashing/lecture/chapter03_hashing_authentication.pdf)
* Week 3 Presentation (video) - Hashing: [here](https://youtu.be/3D11YGD4vFQ)
* Week 3 Presentation (live lecture) - Hashing: [here](https://youtu.be/gh2CI5m2W6Y)

**Lab**

* Unit 3 Lab (PDF): [here](https://github.com/billbuchanan/esecurity/blob/master/unit03_hashing/lab/new_lab03.pdf)
* Unit 3 Lab (video): [here](https://www.youtube.com/watch?v=rnTLr6iUbf0)
* Unit 3 Lab Part 2 (video): [here](https://www.youtube.com/watch?v=FKO6Pjsbp3g)

**Tests**

* Test (Hash Encryption): [here](https://asecuritysite.com/tests/tests?sortBy=cryptobook03)

**Sample Exam Questions**

The following are some sample questions for hashing:

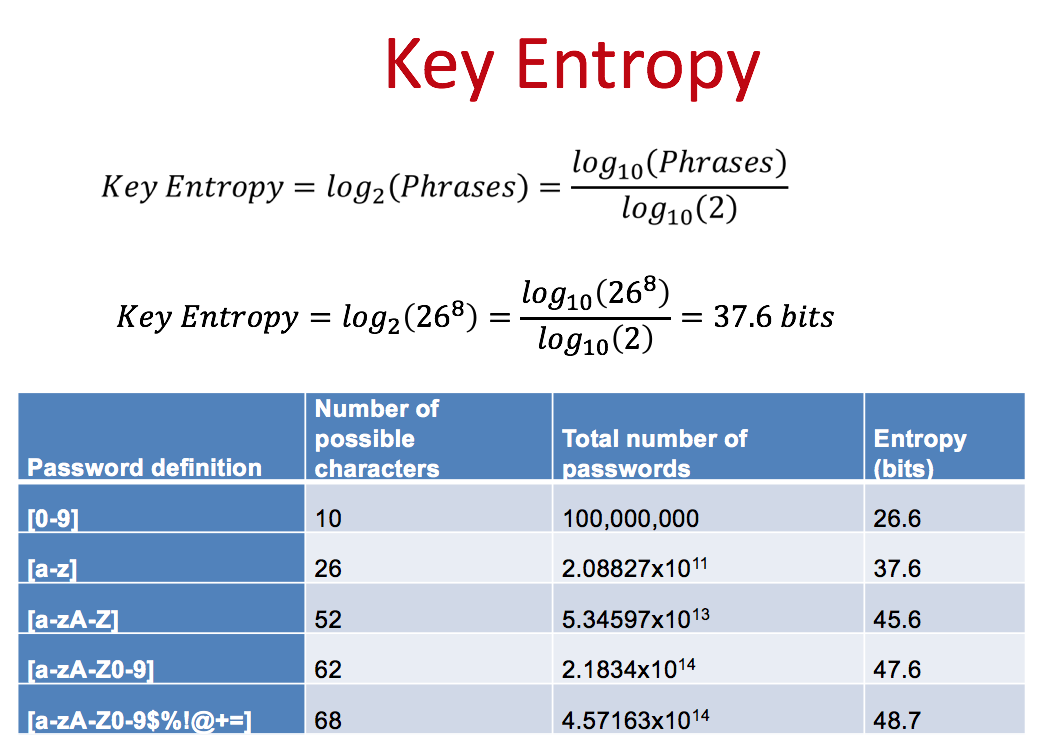
* Outline the importance of storing the salt value with the hashed value when storing hashed passwords.
* Bob is using a password to generate a 128-bit encryption key. Explain why the key space is unlikely to be 2128, and why key entropy could be used to measure the equivalent key size.
  + Where would I find this info? This is related to key enthropy [here](https://asecuritysite.com/encryption/en), and try and understand how key enthopy relates to the strengh of the encryption.
* Bob has just produced a key pair, in a Base-64 format, and now wants to send this to Alice. What advice would you give him on sending the key pair to Alice?
  + Where would I find this info? Have a think about the certificate which is distributed. You can observe it here.
* Bob sends an encrypted message to Alice, and also sends his digital certificate to Alice to prove his identity. How does Alice prove that it is Bob who sent the message?
* Eve has captured a hashed password. How might she use the Cloud to be able to crack the hashed password, and what is a likely too for this?
  + Where would I find this info? This [article](https://www.linkedin.com/pulse/quantum-v-supercomp-cloud-gpu-race-ultimate-cracking-william-buchanan) outlines a number of methods which might be used, included within Cloud cracking.
* Bob is an administrator for a network, and he tells his management team that user passwords are now salted, and they are thus completely secure against attacks. Is he correct? Explain your viewpoint.
  + Where would I find this info? Have a read of the following [article](https://www.linkedin.com/pulse/salting-password-only-secure-when-you-keep-salt-secret-buchanan?forceNoSplash=true).
* Bob looks at the passwd file on his server, and wants to know the type of salting that is used. How would he do this?
  + Where would I find this info? Have a quick look at the additional lab on [Software Hashes](https://asecuritysite.com/lab04_software_hash.pdf). If you can get the Python script to run in Section G, you'll see them all.
* Bob is looking for a new hashing method for storing passwords, and thinks that he will pick the fastest one. Is this a good approach? Explain your answer.
  + Where would I find this info? Think about whether being fast for hashing is a good idea. Have a look at this [article](https://www.linkedin.com/pulse/when-slow-good-great-slowcoach-bcrypt-william-buchanan). But make up your own mind on the subject.
* What are the typical tools that are used to crack hashed password, and what are the methods they will use to crack them?
  + Where would I find this info? Unit 3 and [Lab 2](https://asecuritysite.com/lab03_hashing_and_certs.pdf).
* Why would Eve have an aversion to salt?
* A password is defined as [a-z]. For a four character password, show that there are 456,976 different passwords.
  + Where would I find this info? Have a look [here](https://asecuritysite.com/encryption/passes).
* A password is defined as [a-zA-Z]. For a four character password, show that there are 7,311,616 different passwords.
  + Where would I find this info? Have a look [here](https://asecuritysite.com/encryption/passes).
* A password is defined as [a-zA-Z0-9]. For a four character password, show that there are 14,776,336 different passwords.
  + Where would I find this info? Have a look [here](https://asecuritysite.com/encryption/passes).
* You are working with a security consultant, and he says that you don't need to check the hashing of passwords, as it should work without testing. You disagree with him, and decide to test your hashing method. Initially you must find test vectors for MD5, SHA-1 and SHA-256. Can you find three test vectors, and test them against an on-line calculator?
* At a security presentation a researcher gives a demonstration of Scrypt. In the presentation he shows a demonstration with a password of "password" and fixed salt of "NaCl". For each run he runs the hashing function, the hashed value changes, but, each time, the computation took longer. Which parameter is the researcher likely to be changing, and why does that parameter exist? Can the researcher select any value for the parameter? [Example](https://asecuritysite.com/encryption/scrypt).
* There has been a major data breach within your company, and you are to appear on Sky News to report it. Your company has used PBKDF2 to hash its passwords. How do you explain to your customers that their passwords are unlikely to be breached?
* It was stated in the recent Yahoo hack that:

"We have confirmed, based on a recent investigation, that a copy of certain user account information was stolen from our networks in late 2014 by what we believe is a state-sponsored actor," Lord wrote. "The account information may have included names, e-mail addresses, telephone numbers, dates of birth, hashed passwords (the vast majority with bcrypt), and, in some cases, encrypted or unencrypted security questions and answers."

Do you think the vast majority of the hashed passwords will be cracked? Do you think they had good practice in place for hashed passwords?

**Addendum**

In the lecture, the slide at the end of Unit 2 (Symmetric Key) should be (for an eight character password and with [a-z]):

[](https://camo.githubusercontent.com/19b4e94a8b8a475b384a60007cc577d34b4a8c3b/68747470733a2f2f617365637572697479736974652e636f6d2f7075626c69632f756e697430325f7570646174652e706e67)

**Important points**

* [BCrypt](https://asecuritysite.com/encryption/bcrypt), [PBKDF2](https://asecuritysite.com/encryption/PBKDF2_2) and [Scrypt](https://asecuritysite.com/encryption/Scrypt) are slow hashing methods, which also have salt, and are highly recommended for password storage.
* The strength of the encryption implementation is measured by key entropy. Anything less than 72 bits is likely to be weak.

**e-Security Unit 4: Public Key**

The key concepts are: Basics, RSA, Elliptic Curve and ElGamal.

What you should know at the end of unit?

* + Explain how public key provides both privacy and identity verification.
    - Where would I find this info? This unit explains public key.
  + Understand how the RSA process works, with a simple example.
  + Understand how elliptic curve cryptography works, with a simple example.
  + Explain the operation of PGP.
  + Understands how the private key is used to check the identity of the sender, and how public key is used to preserve the privacy of the message.
  + Explain how the e and d values are determined within the RSA method.
    - Where would I find this info? There are some examples [here](https://asecuritysite.com/log/rsa_examples.pdf).

Presentations

* + Week 4 Presentation (PDF) - Public Key Encryption: [here](https://github.com/billbuchanan/esecurity/blob/master/unit04_public_key/lecture/chapter04_public_msc.pdf).
  + Week 4 Presentation (video) - Public Key Encryption: [here](https://youtu.be/QEYqkxuzoTg).
  + Week 4 Presentation (lecture video - 8 Feb 2020) - Public Key Encryption: [here](https://www.youtube.com/watch?v=PEdCHWdE3zk).

Lab

* + Week 4 Lab (PDF): [here](https://github.com/billbuchanan/esecurity/blob/master/unit04_public_key/lab/new_lab04.pdf)
  + Week 4 Lab (Demo): [here](https://youtu.be/6T9bFA2nl3c)

Public key challenge

* + Bob has the following keys:

-----BEGIN RSA PRIVATE KEY-----  
MIICXgIBAAKBgQDoIhiWs15X/6xiLAVcBzpgvnuvMzHBJk58wOWrdfyEAcTY10oG  
+6auNFGqQHYHbfKaZlEi4prAoe01S/R6jpx8ZqJUN0WKNn5G9nmjJha9Pag28ftD  
rsT+4LktaQrxdNdrusP+qI0NiYbNBH6qvCrK0aGiucextehnuoqgDcqmRwIDAQAB  
AoGAZCaJu0MJ2ieJxRU+/rRzoFeuXylUNwQC6toCfNY7quxkdDV2T8r038Xc0fpb  
sdrix3CLYuSnZaK3B76MbO/oXQVBjDQZ7jVQ5K41nVCEZOtRDBeX5Ue6CBs4iNmC  
+QyWx+u4OZPURq61YG7D+F1aWRvczdEZgKHPXl/+s5pIvAkCQQDw4V6px/+DJuZV  
5Eg20OZe0m9Lvaq+G9UX2xTA2AUuH8Z79e+SCus6fMVl+Sf/W3y3uXp8B662bXhz  
yheH67aDAkEA9rQrvmFj65n/D6eH4JAT4OP/+icQNgLYDW+u1Y+MdmD6A0YjehW3  
suT9JH0rvEBET959kP0xCx+iFEjl81tl7QJBAMcp4GZK2eXrxOjhnh/Mq51dKu6Z  
/NHBG3jlCIzGT8oqNaeK2jGLW6D5RxGgZ8TINR+HeVGR3JAzhTNftgMJDtcCQQC3  
IqReXVmZaeXnrwu07f9zsI0zG5BzJ8VOpBt7OWah8fdmOsjXNgv55vbsAWdYBbUw  
PQ+lc+7WPRNKT5sz/iM5AkEAi9Is+fgNy4q68nxPl1rBQUV3Bg3S7k7oCJ4+ju4W  
NXCCvRjQhpNVhlor7y4FC2p3thje9xox6QiwNr/5siyccw==  
-----END RSA PRIVATE KEY-----

-----BEGIN RSA PUBLIC KEY-----  
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDoIhiWs15X/6xiLAVcBzpgvnuv  
MzHBJk58wOWrdfyEAcTY10oG+6auNFGqQHYHbfKaZlEi4prAoe01S/R6jpx8ZqJU  
N0WKNn5G9nmjJha9Pag28ftDrsT+4LktaQrxdNdrusP+qI0NiYbNBH6qvCrK0aGi  
ucextehnuoqgDcqmRwIDAQAB  
-----END RSA PUBLIC KEY-----

Alice sends him the following ciphered message:

uW6FQth0pKaWc3haoqxbjIA7q2rF+G0Kx3z9ZDPZGU3NmBfzpD9ByU1ZBtbgKC8ATVZzwj15AeteOnbjO3EHQC4A5Nu0xKTWpqpngYRGGmzMGtblW3wBlNQYovDsRUGt+cJK7RD0PKn6PMNqK5EQKCD6394K/gasQ9zA6fKn3f0=

What is the message? You might find some interesting code [here](https://asecuritysite.com/encryption/rsa_example).

* + Bob uses the following parameters for his public key:

RSA Encryption parameters. Public key: [e,N].  
e: 65537  
N: 498702132445864856509611776937010471  
Cipher: 96708304500902540927682601709667939

Can you crack the cipher and find the value, if you know we are using using 60 bit primes [example](https://medium.com/asecuritysite-when-bob-met-alice/cracking-rsa-a-challenge-generator-2b64c4edb3e7)?

A bit of fun

* + A Python program to implement RSA in just 12 lines [here](https://asecuritysite.com/encryption/rsa12).
  + A Python program to crack RSA in just 12 lines [here](https://asecuritysite.com/encryption/rsa12_2).
  + Elliptic Curve methods are used in key handshaking (ECDH). If you want to see the curves that are used click [here](https://asecurity.site/encryption/ecdh3).

**Sample Exam Questions**

The following are sample questions for public key:

* + Bob selects a p value of 7 and a q value of 9, but he cannot get his RSA encryption to work. What is the problem?
  + Bob has selected a p value of 11 and a q value of 7. Which of the following are possible encryption keys: (5,77), (3,77), (9,77), (11,77), and (24,77).
  + Bob and Alice decide to use RSA encryption to send secure email, where Bob uses Alice's public key to encrypt, and she uses her private key to decrypt. What is the main problem caused with this, as apposed to using symmetric encryption?
  + Bob tells Alice that she should send her private key in order that he should encrypt something for her. Outline the main problem caused by this.
  + Security professionals say that RSA keys of over 1,024 bits are secure. What is the core protection against the RSA method being cracked for keys of 1,024 bits and more.
  + Bob says he has had a look at a few RSA public keys and he says that the ones he looked at where all the same. Is he right? If so, what makes public keys different?
  + Research: Netscape had to comply with an export [embargo](https://en.wikipedia.org/wiki/Export_of_cryptography_from_the_United_States) on the size of the keys which can be used for RSA. Which major vulnerabilities have resulted?
  + Bob and Alice get into a debate about the size of the d and e values in the RSA encryption key. Bob says that, in real-life keys, the length of the e value in (e,n) is normally about the same size as the d value (d,n). Alice disagrees. Who is correct?
    - Where would I find this info? Have a look at some practical examples: [Here](https://asecuritysite.com/encryption/rsa2)

Examples

RSA Examples: [here](https://asecuritysite.com/public/rsa_examples.pdf). RSA Keygen: [here](https://asecuritysite.com/encryption/rsa_keygen). ECC Keygen: [here](https://asecuritysite.com/encryption/ecc_keygen).

Quick demos

* + Introduction to RSA: [here](https://www.youtube.com/watch?v=pHES8eNor6k)
  + Introduction to Elliptic Curve: [here](https://youtu.be/_CwIWk6XDmg)
  + Picking the Generator Value (G): [here](https://www.youtube.com/watch?v=-TjSuch3VGU)

*From <*[*https://github.com/billbuchanan/esecurity/tree/master/unit04\_public\_key*](https://github.com/billbuchanan/esecurity/tree/master/unit04_public_key)*>*

**Unit 5: Key Exchange**

The key concepts are: Basics of Key Exchange; Diffie-Hellman, Diffie-Hellman Weaknesses, ECDH, and Passing Key Using Public Key. What you should know at the end of unit?

* + Understand how the Diffie-Hellman process works, with a simple example
  + Understands how the private key is used to check the identity of the sender, and how public key is used to preserve the privacy of the message.
  + Understand the basics of how Bob and Alice generate a shared key with ECC.

Presentations

* + Week 5 Presentation (lecture - Part 1) - Key Exchange: [here](https://youtu.be/l_osjo8r13Q)
  + Week 5 Presentation (lecture - Part 2) - Key Exchange: [here](https://youtu.be/95AuvZNm0Yg)
  + Week 5 Presentation (PDF) - Key Exchange: [here](https://asecuritysite.com/public/unit05_key_exchange.pdf)
  + Week 5 Presentation (class lecture) - Key Exchange (Recorded 14 Feb 2020): [here](https://youtu.be/WIkDh_5198M)

Lab

* + Week 5 Lab (PDF): [here](https://github.com/billbuchanan/esecurity/blob/master/unit05_key_exchange/lab/new_lab05.pdf)
  + Week 5 Lab (Demo): [here](https://www.youtube.com/watch?v=Lnw4FhiOwiU&feature=youtu.be)

Sample Exam Questions

The following are sample questions for key exchange:

* + Eve listens to Bob and Alice's communcication for their Diffie-Hellman handshaking. In order to generate the same key as Bob and Alice, which values will Eve try to determine, and how is it likely to be difficult to gain these?
  + For the following key exchanges, Bob generates x, and Alice generates y. Prove the shared key. [Examples](https://asecuritysite.com/public/diffie_examples.pdf)
    - x=3, y=4, G=4 and N=7. Share=1.
    - x=6, y=15, G=5 and N=23. Share=2.
    - x=5, y=7, G=10 and N=541. Share=193.
    - x=6, y=15, G=5 and N=23. Share=2.
    - x=7, y=7, G=5 and N=11. Share=9.
    - x=7, y=9, G=8 and N=13. Share=5.
    - x=5, y=4, G=2969 and N=9929. Share=8106.
    - x=6, y=5, G=3881 and N=125. Share=792.
    - x=3, y=4, G=3623 and N=1153. Share=939.
  + Why are Forward Security and Ephemeral so important for the security of your keys?

Examples

* + Diffie-Hellman Examples: [here](https://asecuritysite.com/public/diffie_examples.pdf)
  + ECDH Step-by-step: [here](https://asecuritysite.com/encryption/js08)

Quick demos

* + Introduction to Diffie-Hellman: [here](https://www.youtube.com/watch?v=wyNPhNAsmJ0)
  + ECDH [here](https://youtu.be/uQQz3MX-d8I)
  + Picking the Generator Value (G): [here](https://www.youtube.com/watch?v=-TjSuch3VGU)

*From <*[*https://github.com/billbuchanan/esecurity/tree/master/unit05\_key\_exchange*](https://github.com/billbuchanan/esecurity/tree/master/unit05_key_exchange)*>*

* + Bob sends an encrypted message to Alice, and also sends his digital certificate to Alice to prove his identity. How does Alice prove that it is Bob who sent the message?

*From <*[*https://asecuritysite.com/csn11117/unit06*](https://asecuritysite.com/csn11117/unit06)*>*