CAS Cyber Security
A3 Crypto Assignment I
Sample Solution

Document Name:	cas_cyber_sec_crypto_assignment_I_sample_solution_v0.2.docx
Version:	v0.2
Date of Delivery:	November 27th, 2020
Authors:	Cyrill Brunschwiler, Compass Security Schweiz AG
Classification:	TLP RED

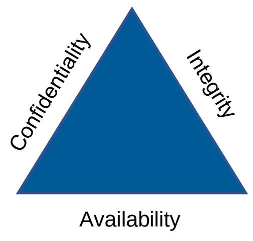
Compass Security Schweiz AG, Werkstrasse 20, Postfach 2038, CH-8645 Jona
T +41 55 214 41 60, team.csch@compass-security.com, www.compass-security.com

Table of Contents

1 A3 Intro and Historical Crypto

This chapter will provide an intro into the foundation of information security, cryptography and a few historical developments.

1. **While most use CIA as the foundation for Information Security some add further words to complete it. Name two more commonly used to enhance the triad?**



Information security's primary focus is the balanced protection of the confidentiality, integrity and availability of data (also known as the CIA triad).

In addition, other properties, such as authenticity, accountability, non-repudiation and reliability can also be involved.

Reference:

- ISO/IEC 27000:2018(E) page 4, https://www.iso.org/standard/73906.html

2. **Cryptography is usually used to achieve or provide a so-called cryptographic service. Name methods or functions (aka cryptographic primitives to achieve the listed services).**

Cryptographic primitives are well-established, low-level cryptographic algorithms that are frequently used to build cryptographic protocols for computer security systems. These routines include, but are not limited to, one-way hash functions and encryption functions.

- One-way hash function to compute a reduced hash value for a message (e.g., SHA-256)
- Symmetric key cryptography to compute a ciphertext decodable with the same key used to encode (e.g., AES)
- Public-key cryptography to compute a ciphertext decodable with a different key used to encode (e.g., RSA)
- Digital signatures to confirm the author of a message
- Mix network to pool communications from many users to anonymize what came from whom
- Private information retrieval to get database information without server knowing which item was requested
- Commitment scheme to commit to a chosen value while keeping it hidden to others, with the ability to reveal it later
- Cryptographically secure pseudorandom number generator

You have been asked to list the primitives to achieve the following services. Thus, we assign them like

Data secrecy

- Private key cryptography
- Public key cryptography

Data integrity

- Digital signatures
- Hash functions

Authentication

- Hash functions/MACs
- Digital signatures

Non-repudiation

- Digital signatures

References

- https://en.wikipedia.org/wiki/Information_security#Key_concepts
- https://en.wikipedia.org/wiki/Cryptographic_primitive#Commonly_used_primitives

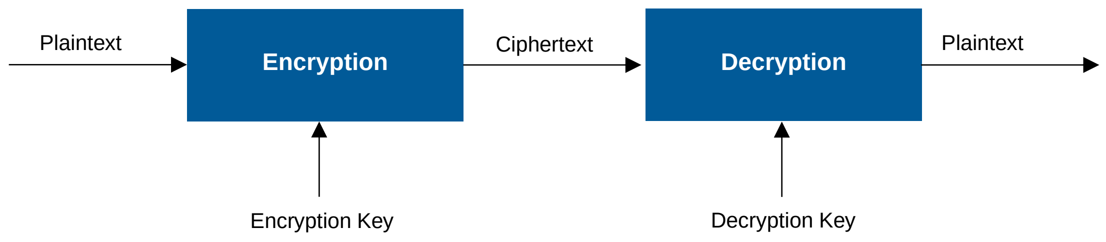
3. **We usually refer to Alice and Bob as parties that exchange information. Could you name three more personas that have a special meaning in crypto? Eg. Mallory for someone that does malicious things.**

Name	Purpose
Alice and Bob	The original, generic characters. Generally, Alice and Bob want to exchange a message or cryptographic key.
Charly	A generic third participant.

Dan	A generic fourth participant.
Eve	An eavesdropper, who is usually a passive attacker
Frank	A generic sixth participant.
Grace	A government representative. Grace may force to implement backdoors or weaken standards.
Ivan	An issuer
Judy	A judge who may be called upon to resolve a potential dispute between participants.
Mallory	A malicious active attacker.
Olivia	An oracle, who provides external data to smart contracts residing on distributed ledger (commonly referred to as blockchain) systems.
Oscar	An opponent, similar to Mallory, but not necessarily malicious.
Peggy	A prover, who interacts with the verifier to show that an intended transaction has actually taken place.
Rupert	A repudiator who appears for interactions that desire non-repudiation
Sybil	A pseudonymous attacker, who usually uses a large number of identities. For example, Sybil may attempt to subvert a reputation system.
Trent	A trusted arbitrator, who acts as a neutral third party.
Trudy	An intruder.
Victor[1]	A verifier, who requires proof from the prover.
Walter	A warden, who may guard Alice and Bob.
Wendy	A whistleblower, who is an insider with privileged access capable of divulging information.

Source: https://en.wikipedia.org/wiki/Alice_and_Bob

4. Draw a basic crypto system and name all inputs, outputs, intermediate products, processes as well as all involved parties above.



References

- Understanding Cryptography: A Textbook for Students and Practitioners, ISBN 978-3-642-04101-3, Figure 1.5

5. Assuming Malory has no access to Alice and Bobs facilities but is in position to interfere with and to snoop on the messages they exchange. Can you point out the difference between active and passive attacks which Mallory could apply? Give examples for both. See https://www.geeksforgeeks.org/active-and-passive-attacks-in-information-security/

A passive attack is the analysis of traffic or data without modification or interference with such. We sometimes call this listening, snoop or eavesdrop on a channel. Passive attacks mainly aim at data stealing or harvesting and keeping it for later processing. Passive attacks are very difficult to detect.

Active attacks include modification, interruption, deferral or replay of data and usually aim to impersonate users, gain illegitimate access or exploit weaknesses to tamper with the confidentiality, integrity or availability of systems and data. Due to the somewhat noisy character of active attacks it is usually possible to detect such by applying monitoring measures at network, protocol and application layers.

6. **Alice and Bob could make use of two major concepts to secure their messages. Either use a symmetric cipher system or a public key cipher system. Explain the difference**

   In symmetric systems the keys for encryption and decryption are the same and must be kept secret by all involved parties.

   In public key systems the encryption key can be shared publicly while the decryption key must be kept secret by the owner. Although, the encryption key is shared publicly, the recipient of the public key must use additional means (certificate authority, other secure channel) to verify the public key really belongs to the other party.

   Symmetric systems have an advantage in performance over public key systems. Many protocols use both cipher systems (hybrid systems) in order to make most out of the two concepts. Thus, combine public shareable keys with speed.

7. **Genius engineers such as Alice are capable to developed their own algorithms to secure messages. Is it a wise decision to keep the algorithms secret? Discuss pros and cons. Get inspired by the Kerkhof's Principle**

   Is it a wise decision to keep the algorithms secret?

Pros	Cons
Yes, it makes it hard for an attacker to guess the algorithm	No, we must assume an attacker to get hold of an encryption or decryption devices sometime.
Yes, the algorithm is unlikely to be affected by published weaknesses in public algorithms	No, it is hard to apply all public studies to the own algorithms
Yes, attackers cannot rely on public effort to break the algorithm but must invest own resources	No, shared effort on algorithm analysis helps to identify and maybe patch or decommission weak algorithms early.
Yes, for symmetric systems the keys need to be kept secret and the problem to keep the algorithm secret too is not much harder.	No, it is hard to ship devices with baked in algorithms and to manage to keep the algorithm secret. It is lots easier to only ship keys.

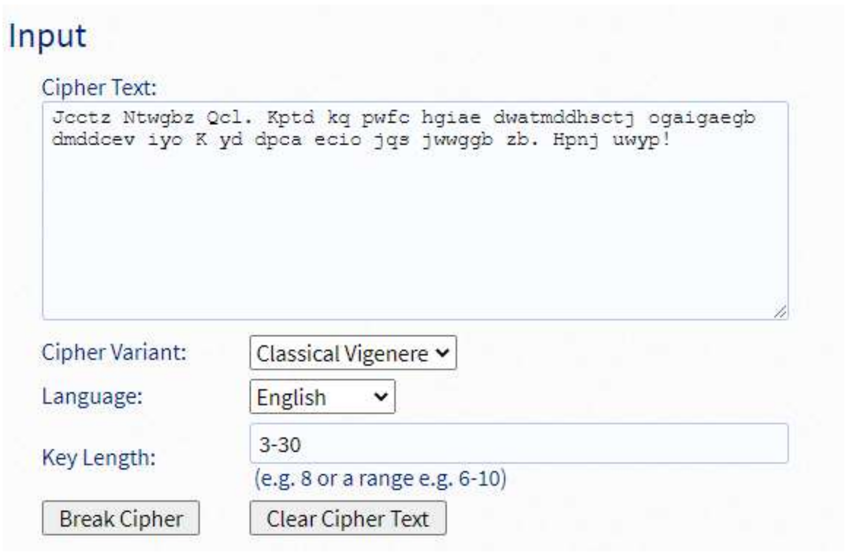
   There might be more reasons - of course.

8. **What you didn't know is that Alice and Bob lived in 1580 and both were fluent in English. They left us a treasure. Unfortunately, all we know about it is a cryptic message "Jcctz Ntwgbz Qcl. Kptd kq pwfc hgiae dwatmddhsctj ogaigaegb dmddcev iyo K yd dpca ecio jqs jwwggb zb. Hpnj uwyp!". Do you know what it means?**

   This exercise needs a bit of OSINT skillz. If you dig into the timeline of classical ciphers you will figure that slightly before 1580 the Vigenere cipher was invented.

   - 1553 - Bellaso invents Vigenère cipher
   - 1585 - Vigenère's book on ciphers

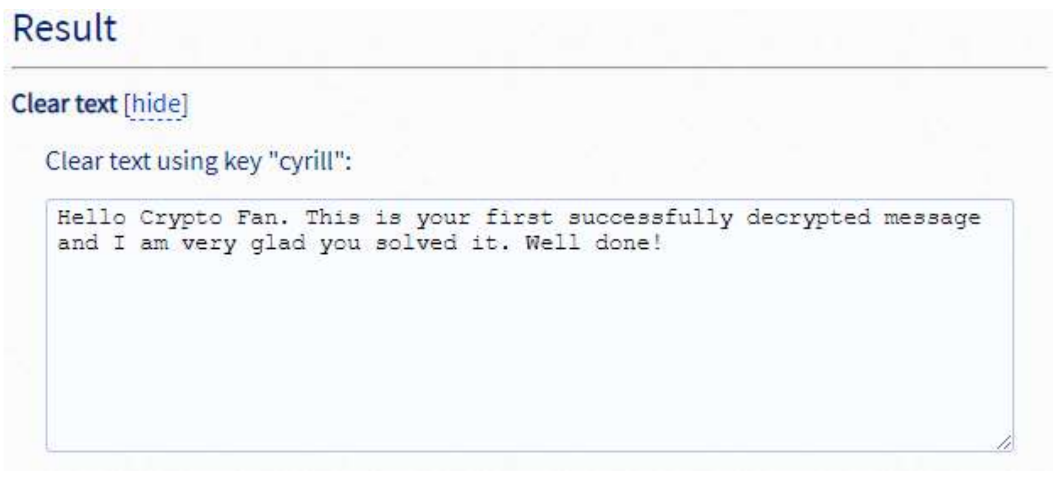
   "both were fluent in English" provides a hint on the plaintext which is likely to be written in their common language. Having that info you search for an online "Vigenere solver" copy the string and break it.


Input

Cipher Text:

```
Jcctz Ntwgbz Qcl. Kptd kq pwfc hgiae dwatmddhsctj ogaigaegb
dmddcev iyo K yd dpca ecio jqs jwwggb zb. Hpnj uwyp!
```

Cipher Variant:	Classical Vigenere
Language:	English
Key Length:	3-30 (e.g. 8 or a range e.g. 6-10)



Result

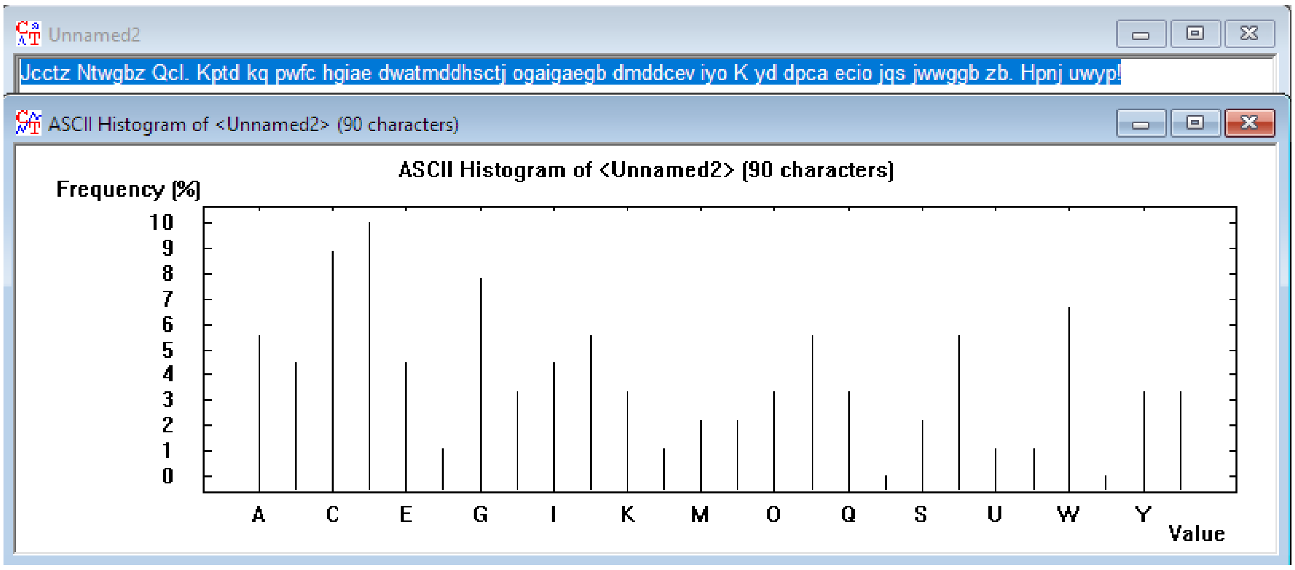
Clear text [hide]

Clear text using key "cyrill":

```
Hello Crypto Fan. This is your first successfully decrypted message
and I am very glad you solved it. Well done!
```

Hello Crypto Fan. This is your first successfully decrypted message and I am very glad you solved it. Well done!

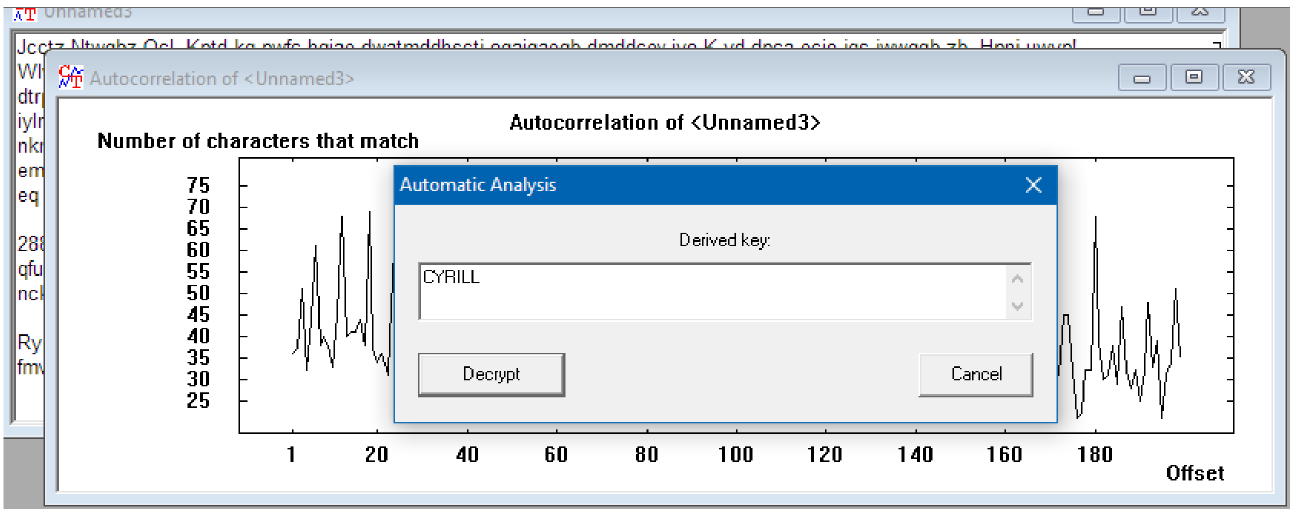
CrypTool cannot solve the ciphertext in exercise 8 due to the limited amount of ciphertext. The frequency analysis does therefore not yield a clear result. Usually you need 200+ characters in order to get reliable results.



Unnamed2
ASCII Histogram of <Unnamed2> (90 characters)
ASCII Histogram of <Unnamed2> (90 characters)
Frequency [%]
10 9 8 7 6 5 4 3 2 1 0
A C E G I K M O Q S U W Y
Value

```
Jcctz Ntwgbz Qcl. Kptd kq pwfc hgiae dwatmddhsctj ogaigaegb dmddcev iyo K yd dpca ecio jqs jwwggb zb. Hpnj uwyp!
```

If you have had given more ciphertext then CrypTool manages to detect the correct key.



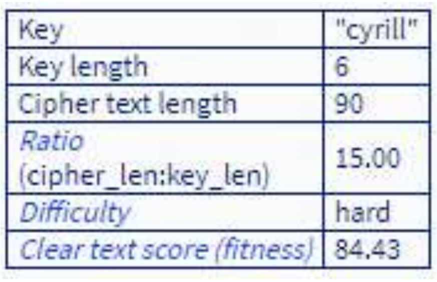
Derived key: CYRILL

References

- https://en.wikipedia.org/wiki/Timeline_of_cryptography
- https://www.guballa.de/vigenere-solver

9. Can you provide details on the algorithm and key in use?

This is a follow-up of question 8 and should be pretty clear now that you solved it.



Key	"cyrill"
Key length	6
Cipher text length	90
Ratio (cipher_len:key_len)	15.00
Difficulty	hard
Clear text score (fitness)	84.43

10. Is a cipher text properly protected if we make sure to have large keys? May you provide an example with the simple substitution cipher?

To answer whether longer keys would help we need to figure the maximum key space for the simple substitution cipher first. The key is composed of the letters of the alphabet: 26 in total. The one we picked is not available for the next, thus its 25 remaining, then 24, 23 and so on, until we are left with 1. To get the maximum possibilities out of this you need to multiply all the numbers or just calculate 26! = 2^{88}.

2^{88} is 88-bits key length and 88-bits security would be quite okay. HOWEVER, the problem with the substitution cipher is buried somewhere else. Attackers can run letter frequency analysis in order determine the plaintext letter and then derive the letter used for the substitution at that very point in text.

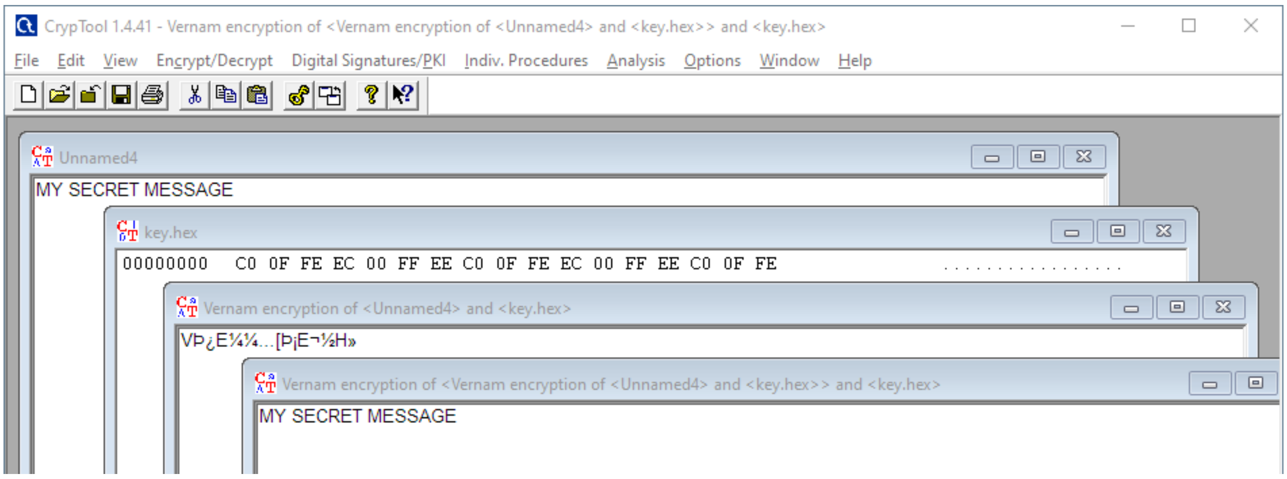
Following that, for the simple substitution ciphers longer keys does not necessarily mean more security- unfortunately.

References
- https://en.wikipedia.org/wiki/Substitution_cipher
- https://en.wikipedia.org/wiki/Letter_frequency

11. If Alice and Bob had chosen to hide the message with a One-Time Pad. Could we decipher it? Name three things they needed to keep attention on to have perfect secrecy and thus to avoid an attacker could recover the plain-text?

- The key must be truly random.
- The key must be at least as long as the plaintext.
- The key must never be reused in whole or in part.
- The key must be kept completely secret.

Spot the problem with the following application of the One-time Pad/Vernam cipher considering the above rules. Solution below the screenshot.


CrypTool 1.4.41 - Vernam encryption of <Vernam encryption of <Unnamed4> and <key.hex>> and <key.hex>
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Unnamed4
key.hex
Vernam encryption of <Unnamed4> and <key.hex>
Vernam encryption of <Vernam encryption of <Unnamed4> and <key.hex>> and <key.hex>

MY SECRET MESSAGE

```
00000000   C0 0F FE EC 00 FF EE C0 0F FE EC 00 FF EE C0 0F FE        .................
```

VÞ¿E¼¼…[Þ¡E¬½H»

MY SECRET MESSAGE

Yes, you are right. The key is not truly random. Even worse, it repeats. And no – it is not double encrypted – OTP is just XOR of the plaintext and the key. If you do this twice you decrypt automatically.

References
- https://en.wikipedia.org/wiki/One-time_pad

2 A3 Modern Ciphers

This chapter will provide an intro into modern cryptography, basic concepts and differences between stream and block ciphers.

1. **Alice and Bob living in 2020 make use of modern crypto systems. You are in the role of Mallory and got hold of some exchanged ciphertexts.**

   1. Can you run a brute-force attack if you know the plaintext is random?
   How would you figure you properly decrypted something if the plaintext you expect is totally random? Every invalid decryption result in something totally random looking. Thus, an exhaustive key search in such scenario is pointless.

   2. Can you run a brute-force attack if you know part of the plaintext but not the algorithm used?
   Theoretically, you may not only guess the key but also the algorithm. Of course this makes it harder. However, the exercise does not mention what key lengths and complexity of algorithms we are talking about. Thus, academically, it is possible. At least you have an indicator what to look for in the plaintext.

   3. Can you run a brute-force attack if you know part of the plaintext and the algorithm used?
   Yes, definitely. This is the very basic scenario for an attack and is applicable to practically all ciphertexts. It is also known as cipher-text only attacks (COA) where the attacker has only access to the ciphertext. For COA one must know something about the plaintext in order to identify successfully decrypted messages. However, that precondition is not to be confused with plaintext-ciphertext pair requirements for known plaintext attacks (KPA).

   References
   - https://en.wikipedia.org/wiki/Attack_model

2. **Symmetric ciphers can be categorized in either stream or block ciphers. Could you**

   1. briefly explain the difference of the two approaches

Stream cipher	Block cipher
Operate on arbitrary length and bit level	Operate on blocks (e.g. 64-bit, 128-bit)
Handle small bursts of data	Better in handling full blocks of data
Keystream is usually XORed with the plaintext	Blocks are crypted one by one and linked using a block mode
Often based on linear feedback shift registers (LFSRs)	Often based on Feistel or substitution/permutation networks

   2. name algorithms as an example for both

Stream cipher	Block cipher
ChaCha, Salsa20 (ARX)	Rijndael (AES winner, SPN)
Grain-128a (LFSR/NLFSR)	Serpent (AES finalist, SPN)
A5/1, A5/2 (LFSR)	Twofish (AES finalist, Feistel network)
RC4 (LFSR)	RC6 (AES finalist, Feistel network)

   Note, that you could create stream ciphers from block ciphers such as using AES in counter mode (CTR). However, I left these intentionally as they still operate on blocks.

   3. discuss speed of stream and block ciphers

   Stream ciphers have a significant speed advantage over block ciphers due to their relatively simple implementation in hardware

4. describe the differences in error propagation

Stream cipher

For flipped bits there is no error propagation for synchronous or self-synchronizing stream ciphers.

If single bits get flipped during transmission then the very same bit in the plaintext get flipped too.

1111111111111111	Plaintext
0101010101010101	Key applied
1010101010101010	Ciphertext
==>	Transmission
1011101010101010	Erroneous CT
0101010101010101	Key applied
1110111111111111	Erroneous PT

If bits get lost or added during transmission then all following bits get scrambled.

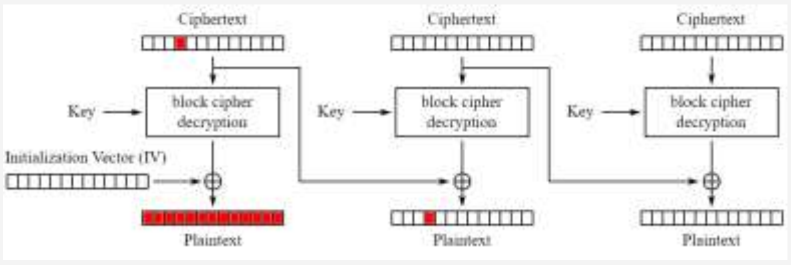
1111111111111111	Plaintext
0101010101010101	Key applied
1010101010101010	Ciphertext
==>	Transmission
10101101010101010	Added bit
0101010101010101	Key applied
1111000000000000	Erroneous PT

Self-synchronizing stream ciphers add features that help to recover of such scenario.

Block cipher

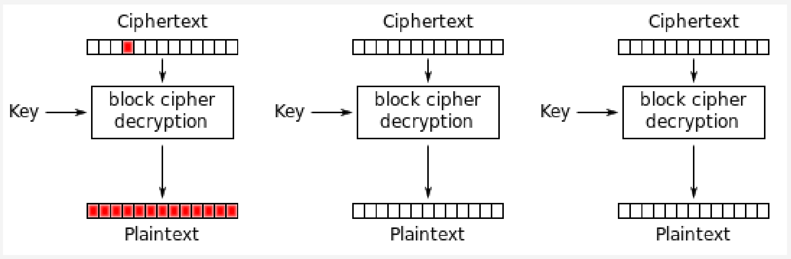
Depending on the block mode, errors in transmission propagate to one or more blocks in the plaintext.

E.g. for CBC mode decryption, assuming byte 4 got transmission errors



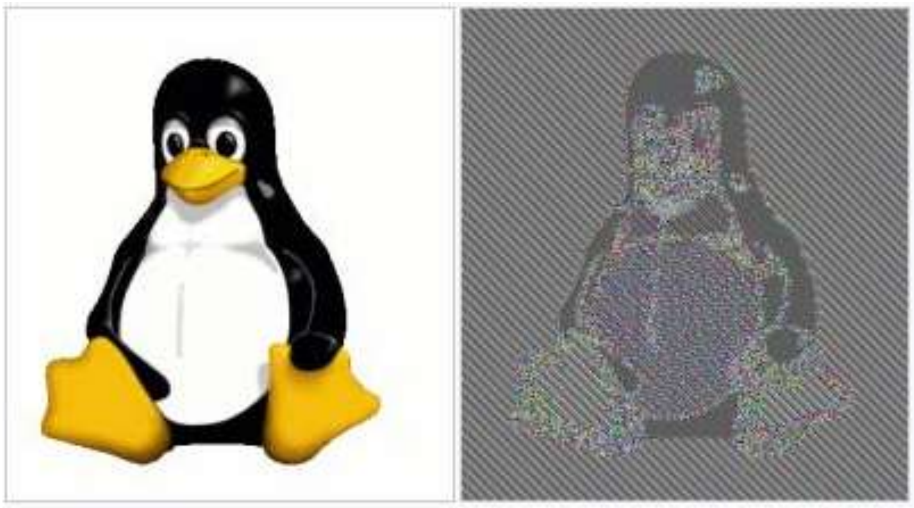
In CBC mode a bit error propagates over to plaintext blocks.

In ECB mode only a single block would be affected.



5. explain the term "message dependency"

Electronic Code Book (ECB) mode of operation for example is message independent. The problem using ECB becomes clear if you encrypt an image. Due to the same plaintext/ciphertext pairs the structure of the image remains in the ciphertext.



We actually would rather like to have different ciphertexts for same plaintexts to avoid Tux being exposed 😉 and thus would prefer "message dependency". Message dependency not only defeats dictionary attacks but frequency analysis in general. Moreover, it complicates the manipulation of encrypted ciphertexts.

E.g. CBC and CFB mode add message dependency, ECB and OFB mode are message independent.

References
- https://en.wikipedia.org/wiki/Stream_cipher
- https://en.wikipedia.org/wiki/Block_cipher

3 A3 Block Ciphers

This chapter will discuss properties of block ciphers - its design, modes, behavior and requirements.

1. The design of block ciphers must achieve confusion and diffusion. Please explain the two terms.

Confusion is the complex relationship between the key and the ciphertext (e.g. non-linearity). Practically, every single bit of ciphertext should depend on large parts of the key. Confusion ultimately aims that one cannot determine the key from the ciphertexts.

Diffusion spreads the influence of the plaintext to the ciphertext. Meaning, small changes in the plaintext results in equal changes over all ciphertext bits. Diffusion assures one cannot determine the plaintext from the ciphertext.

References
- Cryptography and Network Security, ISBN 978-0133354690

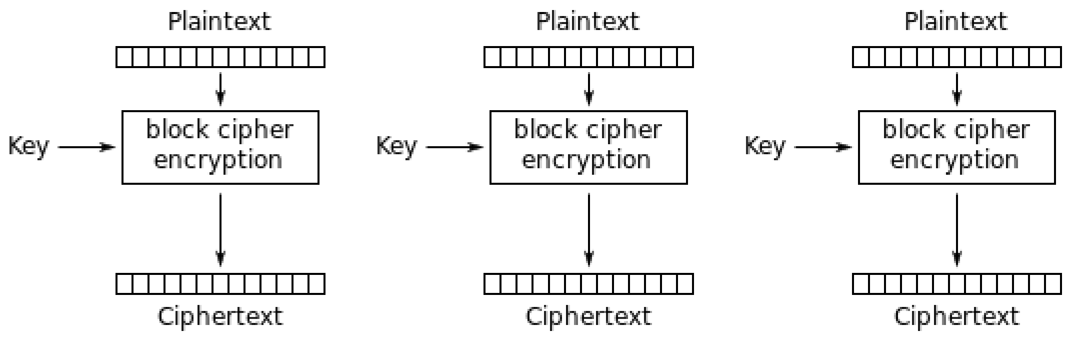
2. **Name two approaches to design a block cipher and give examples of algorithms for each**

Feistel Network	Substitution/Permutation Network (SPN)
DES	Rijndael
Twofish	Serpent
RC6	Kuznyechik

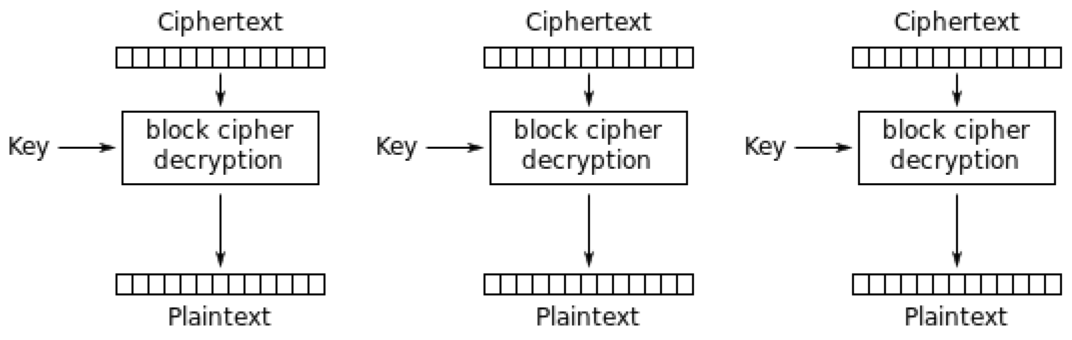
3. Block ciphers use a block mode to encrypt larger portions of plaintext. I need you to compare these block modes: ECB, CBC, CTR, GCM, EAX.

1. **study the scheme that outlines encryption and decryption, naming all paths, inputs, outputs and components**
2. **provide details on their behavior (message dependence, IV handling, error propagation, padding requirements)**
3. **name problems that could arise when wrongly using the mode.**

ECB Mode of Operation: Encryption



ECB Mode of Operation: Decryption



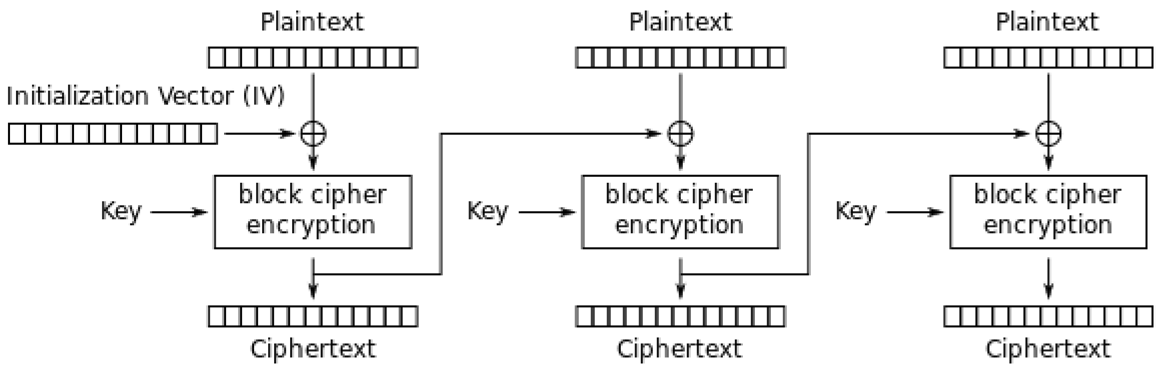
ECB Mode of Operation: Properties

IV handling	ECB has no initial vector
Message dependence	ECB is message independent as block are handled independently
Error propagation	Error propagation is limited to a single block. 1 bit to 1 block
Padding requirements	ECB cannot handle plaintexts shorter than a block. Thus, padding is required

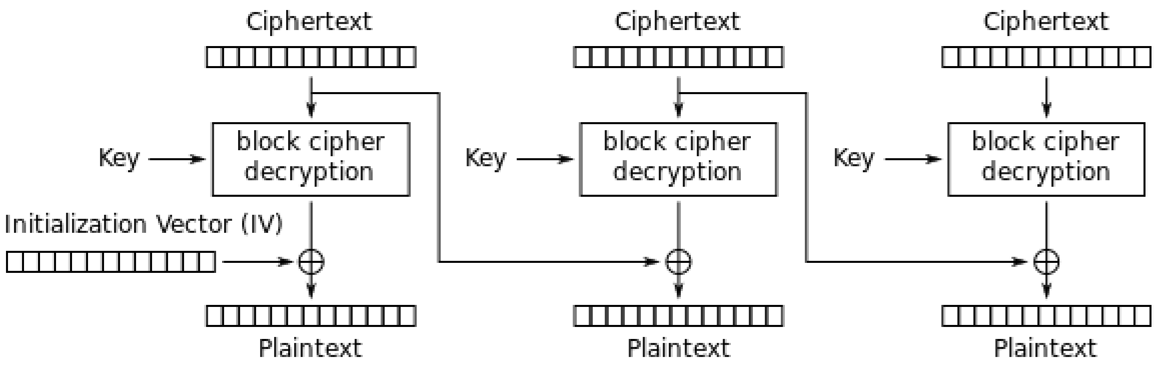
ECB Mode of Operation: Problems

ECB mode ciphertexts are prone to the frequency analysis and dictionary attacks. Theoretically, ECB could be used "securely" under certain circumstances. Either assure all encrypted plaintexts differ or create a new key for every new plaintext. Specifically, the former is not very practical and therefore ECB should be avoided at all.

CBC Mode of Operation: Encryption



CBC Mode of Operation: Decryption



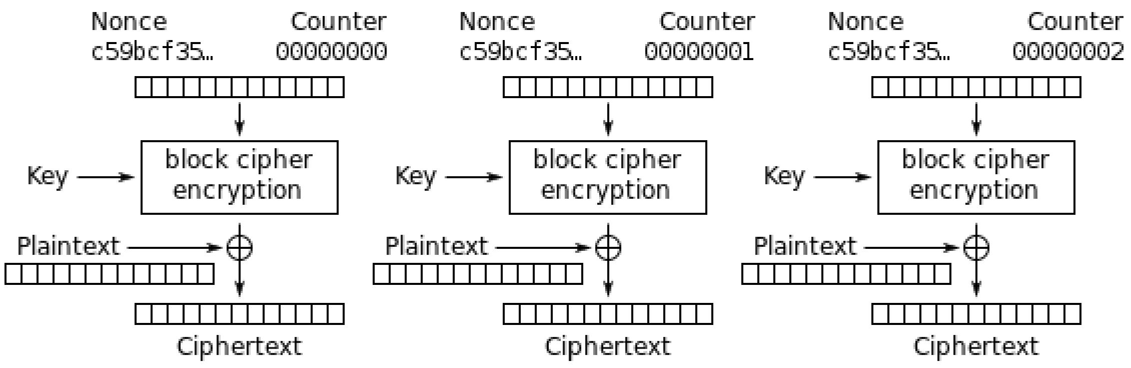
CBC Mode of Operation: Properties

IV handling	The initial vector should be unpredictable and random
Message dependence	CBC adds message dependency by XORing previous ciphertext blocks
Error propagation	Errors in a single ciphertext block propagate over two blocks of plaintext at maximum
Padding requirements	CBC cannot handle plaintexts shorter than a block. Thus, padding is required

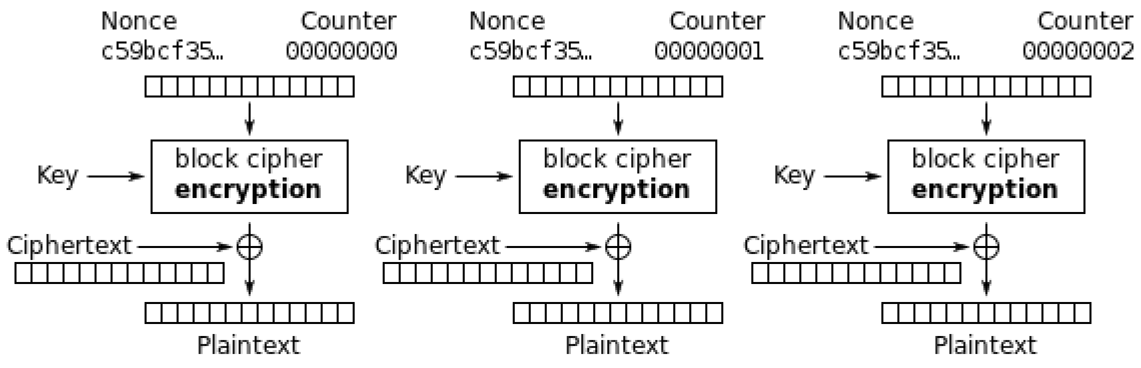
CBC Mode of Operation: Problems

Without proper application the CBC mode is prone to chosen ciphertext attacks. It is vital to either choose ephemeral keys or use a random, unpredictable initial vector for every new message. Anyways, depending on the application behavior, CBC may still be prone to padding oracles as the mode does not include integrity protection. Thus, you are best advised to bet on an AEAD scheme instead of relying on CBC.

CTR Mode of Operation: Encryption



CTR Mode of Operation: Decryption



Note that decryption is practically the same as encryption in CTR mode whereby to only difference is that either the plaintext or ciphertext gets XORed with the encrypted counter.

CTR Mode of Operation: Properties

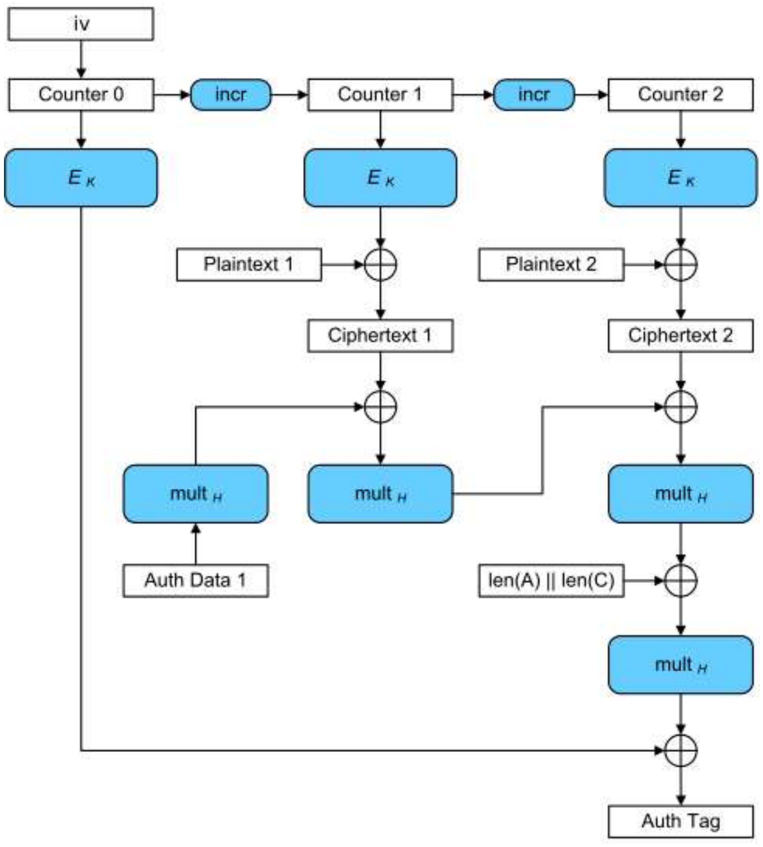
IV handling	Well, there is no IV but there needs to be a nonce
Message dependence	CTR mode encrypts block by block and thus is message independent
Error propagation	Errors in the ciphertext propagate in the very same block for the very same bits only.
Padding requirements	CTR mode can handle arbitrary length of plaintext and does not need padding.

CTR Mode of Operation: Problems

The construct basically forms a stream cipher built from a block mode. Thus, the very same rules apply as for stream ciphers. Following that, the keystream must never repeat and different keystream for every message should be used. To achieve this CTR mode should use nonces and take care the counter does not overflow.

The mode does not provide integrity protection. Thus, you are best advised to bet on an AEAD scheme instead of relying on CTR mode.

GCM Mode of Operation: Encryption / Decryption, Authentication



As you might notice, Galois Counter Mode (GCM) relies on the CTR mode and let us assume similar properties and problems apply. The major difference compared to the CTR mode is the MAC construct that allows parallel calculation of the authentication tag which is used to ensure integrity in the same go.

Note that GCM mode may also integrity protect and tie associated data (Auth Data 1) to the ciphertext. Assuming you have some protocol that has plaintext and ciphertext parts then GCM allows to integrity protect the entire message. So, attackers could not swap the plaintext or ciphertext for something else.

GCM Mode of Operation: Properties

IV handling	The IV requires to be unique for each new encryption under the same key
Message dependence	GCM mode encrypts block by block and thus is message independent
Error propagation	Errors in the ciphertext propagate in the very same block for the very same bits only.
Padding requirements	GCM mode can handle arbitrary length of plaintext and does not need padding.

GCM Mode of Operation: Problems

It is important to note that GCM requires unique IVs for each new encryption under the same key to remain secure and sufficiently long authentication tags should be chosen.

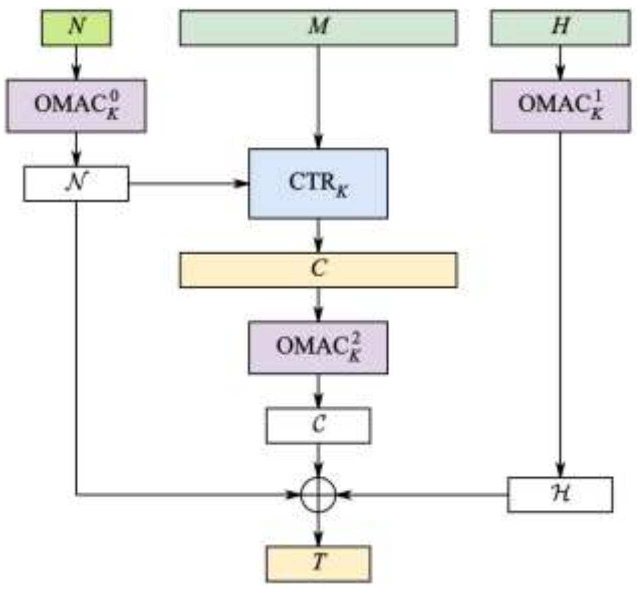
Systems that use GCM should restrict the number of invalid authentication tag guesses under a specific key.

GMAC is a message authentication only construct based on the GCM mode. However, have started to identify various weaknesses. Thus, you are advised not to use GMAC.

References

- https://en.wikipedia.org/wiki/Galois/Counter_Mode
- https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf
- Saarinen, "Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes". FSE 2012.

EAX Mode of Operation : Encryption / Decryption, Authentication



EAX Mode of Operation is two-pass scheme and thus much slower than GCM. However, EAX mode of operation is an on-line mode. Thus, you may start encryption before the whole plaintext is known which is especially interesting for very large messages or streams.

As with GCM, EAX can protect additional plaintext by including the optional authentication header H into the authentication tag T generation.

EAX Mode of Operation : Properties

IV handling	EAX is nonce based. So, there is no OV
Message dependence	EAX mode encrypts block by block and thus is message independent
Error propagation	Errors in the ciphertext propagate in the very same block for the very same bits only.
Padding requirements	EAX mode can handle arbitrary length of plaintext and does not need padding.

EAX Mode of Operation: Problems
- EAX mode foresees a nonce and as the name implies it should be used only once.
- There is a modification of EAX known as EAXprime which can be easily broken for messages shorter than the key.

References
- https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
- https://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf
- https://en.wikipedia.org/wiki/Padding_(cryptography)#Block_cipher_mode_of_operation
- Attacks and Security Proofs of EAXprime, https://eprint.iacr.org/2012/018.pdf

4. What do you do with plaintext that do not fit a multiple of the block size?

Block modes that are some sort of stream cipher construct like OFB, CFB, CTR; CCM, CWC, GCM or EAX do not need padding as they can handle arbitrary length messages. However, those modes that operate at the block level like ECB or CBC need a padding to make the last part of the message fit into a multiple of blocks.

Padding can be done either at the bit or byte level and there are various approaches and standards. E.g. PKCS#7 would fill 5 missing bytes with 05 05 05 05 05 or 7 missing bytes with 07 07 07 07 07 07 07.

It is important that systems who check the padding of messages to not report padding success or failure to the sender as the sender could use the information to recursively recover plaintext bits from the ciphertext. The attack is known as "padding oracle" and was first discovered by Vaudenay at EPF Lausanne. Specific attacks have been developed and applied to SSL and TLS including "Lucky 13 attack" by Kenny Paterson at Royal Holloway University of London – nowadays professor at ETH Zurich.

References
- https://en.wikipedia.org/wiki/Padding_(cryptography)
- PKCS #7, https://tools.ietf.org/html/rfc5652
- Lucky 12, RHUL, http://www.isg.rhul.ac.uk/tls/TLStiming.pdf
- Padding Oracle, Vaudenay, EPFL, https://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf

4 A3 Public Key Algorithms

This chapter deals with public key algorithms and hard problems. You will also practice a bit with popular algorithms. It is crucial to understand public-key algorithms as they are the foundation of basically all hybrid communication schemes, digital signatures and various applications. E.g. PGP, S/MIME, TLS, IPSEC, x.509 Certificates, Mobile App Signing to name a few popular. Very large parts of our digital life depend on it.

1. **Symmetric ciphers rely on shared secrets and thus lack the convenience of key distribution that comes with public key algorithms. Can you think of problems that still remain unsolved?**

   In public key systems, the public key will be shared to peers. The problem arises when Mallory manages to replaces the public key with its own copy on the way to the peers. Mallory will then be in the position to decrypt the message and re-encrypt it with the original public key. A man-in-the-middle situation. This can be countered if either receivers use a secure second channel to check with the sender or by introducing a trusted third party (PKI, CA) that certifies the public key belonging to the originator.

   References
   - https://en.wikipedia.org/wiki/Public-key_cryptography#Alteration_of_public_keys

2. **The class of public key algorithms make use of so-called "hard problems" to assure that private keys cannot be calculated from public keys. The problems are basically a kind of one-way functions. Easy to calculate in one direction but hard to reverse. Describe the two popular "hard problems" in simple words and give an example of an algorithm or protocol that makes use of it.**

   Integer factorization
   - Given a number n being the product of two large prime numbers, it is considered very hard to find the two primes as there is no efficient algorithm for this.
   - The RSA algorithm relies on the construct of the multiplication of two large primes.

   Discrete logarithm problem (DLP)
   - Assuming we have been given y, g and p it is very hard to find the exponent n for $y = g^n \bmod p$. For small p this is of course a quick trial and error exercise. You may want to try it for example with $9 = 2^n \bmod 11$. However, for large primes p, there is no efficient algorithm to calculate n.
   - ElGamal and Diffie-Hellmann base on the DLP.

   References
   - https://en.wikipedia.org/wiki/Computational_hardness_assumption#Common_cryptographic_hardness_assumptions

3. **RSA is by far the most popular public key algorithm. That said, you are expected to describe the key setup (components and criteria) as well as the encryption and decryption formulas. Make a simple example with very small numbers.**

   1. Choose p and q where both are prime numbers. Usually large random ones. p=29, q=37
   2. Calculate n as the product of two primes. n = p * q = 29 * 37 = 1073
   3. Choose e to be larger than 1 and smaller than n. Usually e is chosen 3, 2^4+1, 2^{16}+1. We choose e=17
   4. Calculate (p-1) * (q-1) = 28 * 36 = 1008
   5. Calculate d as multiplicative modular inverse of n mod (p-1) * (q-1). $d \equiv e^{-1} \bmod (p-1) * (q-1) \equiv 17^{-1} \bmod 1008 \equiv 593$

   Our key material is
   - Public key (n,e) = (1073,17)
   - Private key (d) = (593)

   Encryption example
   1. Message m = **14**
   2. Ciphertext $c = m^e \bmod n = 14^{17} \bmod 1073 = 214$

   Decryption example
   1. Message $m = c^d \bmod n = 214^{593} \bmod 1073 =$ **14** 🥳

   References
   - https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Operation)
   - https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
   - https://planetcalc.com/3311/ (modular inverse calculator)

4. **Do the same for ElGamal.**

1. Choose shared prime p = 37
2. Choose shared generator g whereby 2 <= g <= p - 2. g = 13
3. Choose private key x where x is randomly chosen number that fits 1 <= x <= p-2. x = 4.
4. Calculate public key y = g^x mod p = 13^4 mod 37 = 34

Our key material is

-	Shared params (p,g)	= (37,13)
-	Public key (y)	= (34)
-	Private key (x)	= (4)

Encryption example

1. Message m=**12**
2. Random k=14
3. Calculate g^k mod p = 13^{14} mod 37 = 25
4. Calculate c = m * y^k mod p = 12 * 34^{14} mod 37 = 192
5. Encrypted message e = (g^k, c) = (25, 192)

Decryption example

1. Note -x = (p - 1 - x) mod p
2. Calculate $(g^k)^{-x}$ mod p = 25^{32} mod 37 = 7
3. Message m = c * $(g^k)^{-x}$ mod p = 192 * 7 mod 37 = **12** 🥳

References

- https://en.wikipedia.org/wiki/ElGamal_encryption#The_algorithm

5. **Alice sent Bob an encrypted number: 587. Mallory has intercepted the message and also holds a copy of Bob's RSA public key (n,e 2773,17). Proof that small hard problems are not hard and calculate the plaintext number.**

In order to decrypt the message, we need the private key d which is calculated as d ≡ e^{-1} mod (p-1) * (q-1). Thus, we need to guess the primes p and q. Actually, we have n which is n = p * q. Either make it a pen and paper exercise or fall back to an online factorization calculator to get p and q. Anyways, you will figure 2773 = 47 * 59.

d ≡ e^{-1} mod (p-1) * (q-1) ≡ 17^{-1} mod (47-1) * (59-1) ≡ 17^{-1} mod (47-1) * (59-1) ≡ 17^{-1} mod 2668 ≡ 157

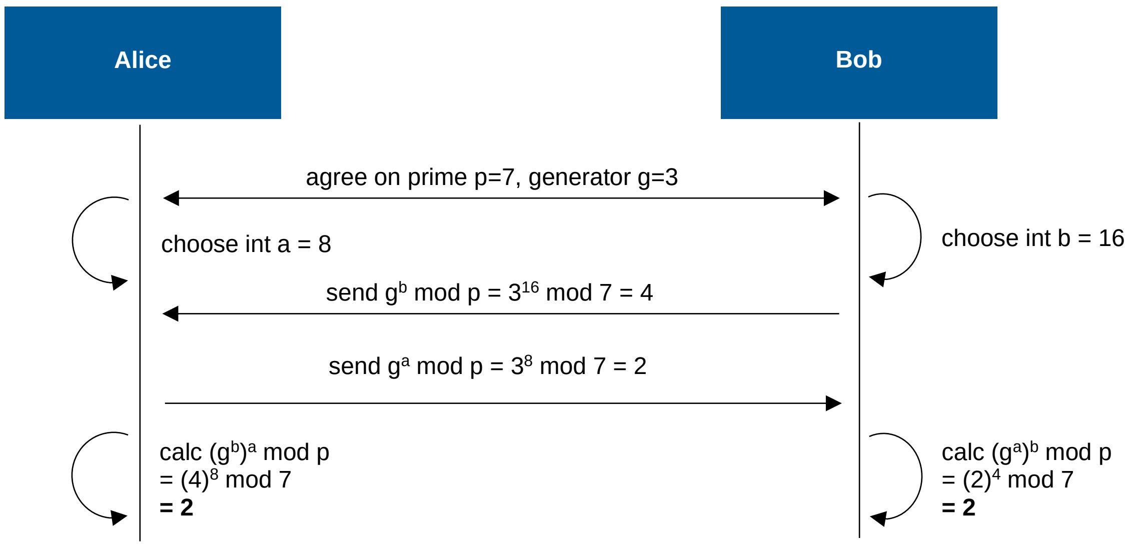
m = c^d mod n = 587^{157} mod 2773 = **31** 😎

References

- https://en.wikipedia.org/wiki/List_of_prime_numbers#The_first_1000_prime_numbers
- https://lcmgcf.com/factors-of-2773/ (factorization tool)
- https://planetcalc.com/3311/ (modular inverse calculator)

6. **The Diffie-Hellman key agreement protocol is an early follower of the public key algorithm idea.**

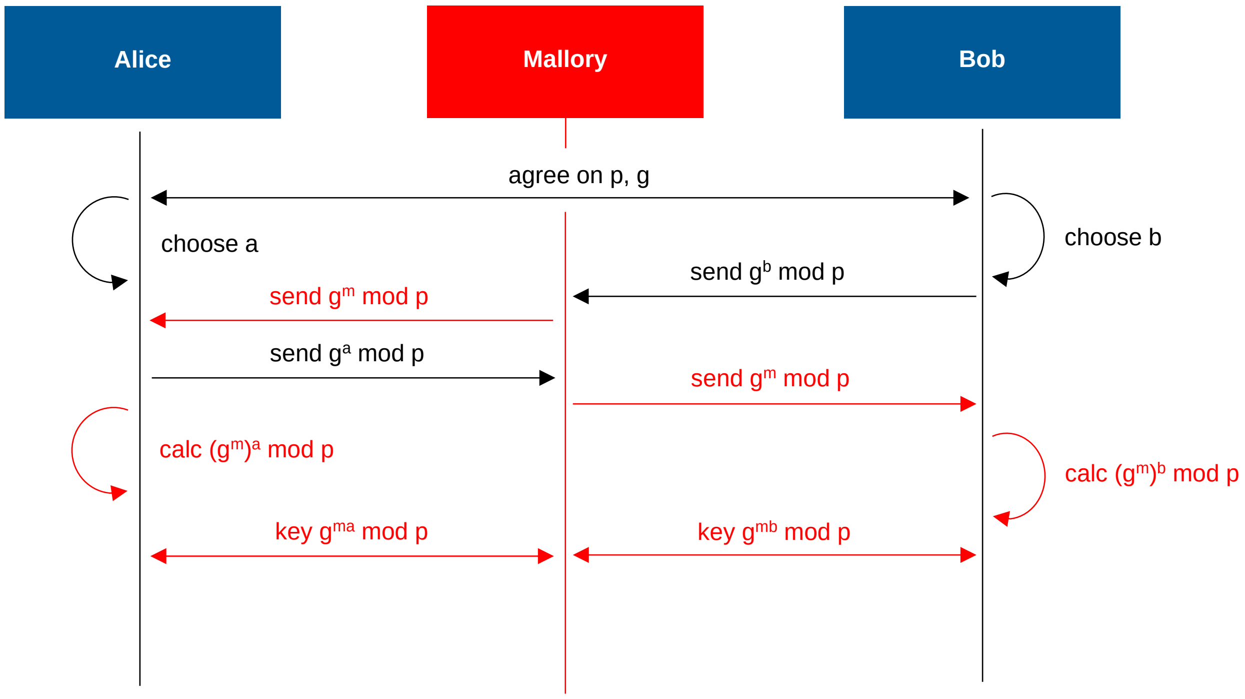
   1. **How does the protocol work? Use Alice and Bob as parties.**


Alice
Bob
agree on prime p=7, generator g=3
choose int a = 8
choose int b = 16
send g^b mod p = 3^{16} mod 7 = 4
send g^a mod p = 3^8 mod 7 = 2
calc $(g^b)^a$ mod p = $(4)^8$ mod 7 = 2
calc $(g^a)^b$ mod p = $(2)^4$ mod 7 = 2

2. **What is the public and what is the private key composed of?**

   There are no public and private keys. Diffie-Hellmann is a key agreement protocol to agree on a symmetric key based on the DLP.

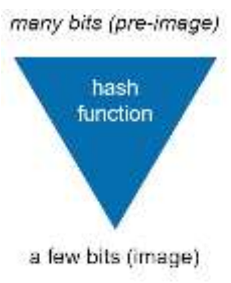
3. **Assuming Mallory is in the position to rely and alter communication. Could you describe an attack that may allow interception of traffic?**



5 A3 Hash Functions

This chapter will discuss important aspects of hash functions and related attacks. We use hash functions as a base to proof integrity of communications or files but hashes are also often used to avoid storage of plaintext passwords in web application databases or the Active Directory. It is important to understand how hashes work in order to provide solid guidance for implementation or judge whether a hash is applied the right way.

1. Hash functions create fixed length representations from arbitrary length inputs. When we usually call something a hash we are loosely referring to what academia is calling the "image". Study a basic hash function chart, its labels, all inputs, outputs and processes.



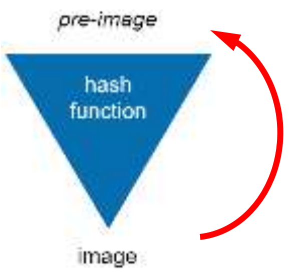
Copy of course slides.

2. What are hash functions good for? List five use cases or typical applications.

- Check data integrity (files, memory, messages)
- Signature generation and verification
- Password checking and storage
- Proof-of-work (e.g. crypto currencies)
- Unique identifiers

3. With hash functions come three important properties: collision resistance, pre-image resistance and second pre-image resistance. Describe the three based on and give hints on which is the easiest to exploit.

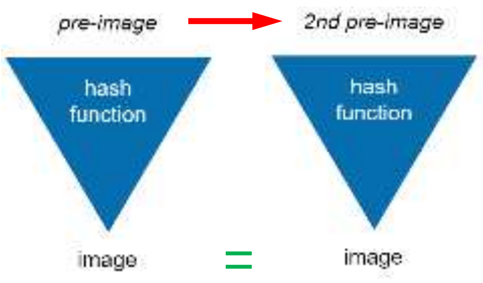
Pre-Image Resistance



The property basically means that it is hard to invert the hash function.

Given an image, it is infeasible to find any pre-image such that H(pre-image) = image.

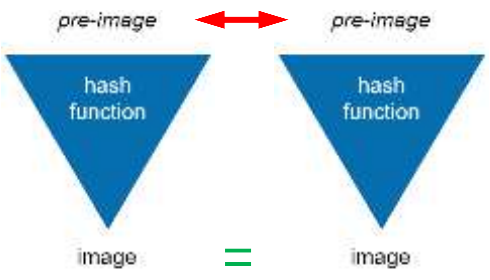
Second Pre-Image Resistance



Generally speaking, the property means that it is hard to find a second document that yields the same hash as for an existing document/hash combination.

Speaking in functions… given a pair (pre-image_a, image_a) it is infeasible to find another pre-image_b where H(pre-image_b) = image_a.

Collision Resistance



Collision resistance implies that it is hard to find two inputs that result in the same output. Or stated as a function: It is hard to find two different pre-images a and b such that H(pre-image a) = H(pre-image b).

Note that compared to the 2nd pre-image resistance both inputs for this attack can be tweaked and thus the birthday paradox applies. Therefore, the computational effort is always $2^{n/2}$. Following that and except for broken functions, this is the most efficient attack against a hash function.

4. Assuming we have a perfect hash function that produces 128-bit output (Same size as with MD5). How resistant is the said function against collisions? Can you calculate the computational effort and would this be sufficient protection nowadays?

E.g. MD5 digest size is n=128 bits. Provided the function has no other deficiencies than the effective security claim for collisions is $2^{n/2} = 2^{128/2} = 2^{64}$

Resistance against collisions is for all hash functions half of their effective strength as the birthday paradoxon applies to all. Anyways, ECRYPT II CSA recommends:

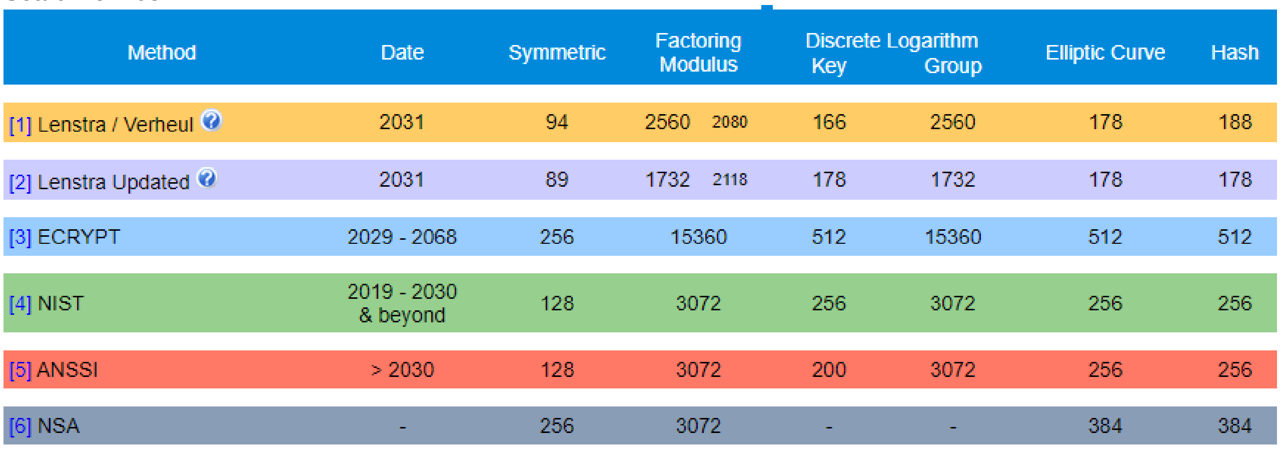
"Hash function outputs should be, in our opinion, a minimum of 160 bits in length for legacy applications and 256 bits in length for all new applications. Hash functions are probably the area of cryptography which has had the most attention in the past decade. This is due to the spectacular improvements in the cryptanalysis of hash functions, as well as the subsequent SHA-3 competition to design a replacement for our existing set of functions."

See https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf, February 2018

5. If we need a secure hash function to assure proper integrity protection for the next decade. What would be minimal size for a good function? Check https://www.keylength.com/ and argue your assumptions and recommendation.

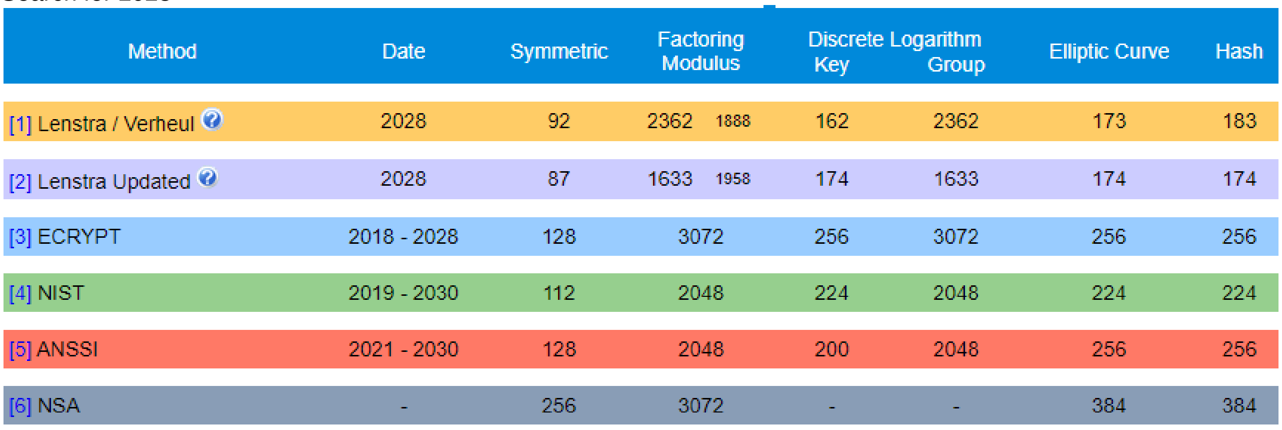
Assuming we are in 2021 yet. The next decade means until at least 2031.

Search for 2031



Method	Date	Symmetric	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
[1] Lenstra / Verheul	2031	94	2560 2080	166	2560	178	188
[2] Lenstra Updated	2031	89	1732 2118	178	1732	178	178
[3] ECRYPT	2029 - 2068	256	15360	512	15360	512	512
[4] NIST	2019 - 2030 & beyond	128	3072	256	3072	256	256
[5] ANSSI	> 2030	128	3072	200	3072	256	256
[6] NSA	-	256	3072	-	-	384	384

Search for 2028



Method	Date	Symmetric	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
[1] Lenstra / Verheul	2028	92	2362 1888	162	2362	173	183
[2] Lenstra Updated	2028	87	1633 1958	174	1633	174	174
[3] ECRYPT	2018 - 2028	128	3072	256	3072	256	256
[4] NIST	2019 - 2030	112	2048	224	2048	224	224
[5] ANSSI	2021 - 2030	128	2048	200	2048	256	256
[6] NSA	-	256	3072	-	-	384	384

ECRYPT suggests hashes with security for 2031 with 512 bits which is an outlier as they double in 2028. NSA surely adds some margin too. Anyways, comparing ANSSI, NIST and ECRYPT until 2028 they are quite close. If we just interpolate this a bit for 2031 then we could conclude:

It is probably safe to say that hashes with security of 256 bits are safe to use until 2031.

References
- https://keylength.com

6 A3 Message Authentication Codes

This chapter will teach you how secure message authentication codes are being created and why it is a quite bad idea to build some own MAC bluntly. Moreover, the chapter will ask for aspects of authenticated encryption. MACs are an integral part of everyday use in file "encryption" or communication "encryption". Usually we call it "encryption" but imply that also the integrity is assured and this is where MACs play a significant role. Moreover, MACs can be used to ensure a message was sent by a specific entity and thus MACs can be used for authentication purpose.

1. **MACs aim to cryptographically secure message integrity and are either based on hash functions or block ciphers. They make use of symmetric keys for generation and verification. Provide reasons why it is a bad idea to design an iterated hash mac like**

   1. mac = hash(concat(key, message))
   Method 1 is also known as prefix method and is unfortunately vulnerable to so called length extension attacks.

   "An attacker may append any blocks to the message and update the MAC accordingly, using the old MAC as the initial chaining variable" [1].

   Note, that not all hash functions are susceptible to length extension attacks. It depends on the underlying hash design. The Keccak hash function (SHA-3 winner) for example is immune. It's specifically a problem with iterated hash functions such as those based on the Merkle–Damgård construction.

   2. mac = hash(concat(message, key))
   Method 2 is known as suffix method and vulnerable to attacks related to the birthday paradox.

   "A concern with the secret suffix method is that an on-line collision attack on the hash function may be used to obtain an internal collision" [1].

   3. mac = hash(concat(key, message, key))
   Method 3 is known as envelope method and vulnerable to divide-and-conquer attacks.

   "It has been claimed" … "that a divide and conquer attack (or partitionable attack) against K1 and K2 is not possible, and that breaking this method requires exhaustive search for a key of k1 + k2 bits. This statement is now shown to be false." [1].

   References

   [1] On the Security of Iterated Message Authentication Codes, https://cr.yp.to/bib/1999/preneel.pdf

   - https://people.scs.carleton.ca/~paulv/papers/Euro96-2MACs.pdf
   - https://en.wikipedia.org/wiki/HMAC#Design_principles
   - https://en.wikipedia.org/wiki/Length_extension_attack
   - https://en.wikipedia.org/wiki/Birthday_problem

2. **Explain the HMAC construction and point out advantages over the above approaches.**

   The HMAC construction requires two constants the ipad and opad, each 64 bytes.

ipad	0x36 ……………… .0x36
opad	0x5c ………………..0x5c

   Moreover, the key K is expanded with zeros to fit the 64 bytes.

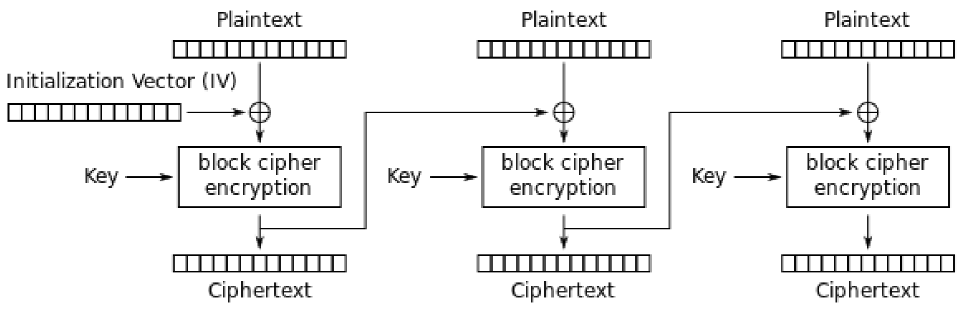
   mac = HMAC(K XOR opad, H(K XOR ipad, text))

   The HMAC construct was specifically motivated due to the attacks outlined in 1.1 – 1.3. For example, the inner hash reduces the text to a fixed length output which defeats length extension attacks.

   References
   - https://en.wikipedia.org/wiki/HMAC
   - https://tools.ietf.org/html/rfc2104

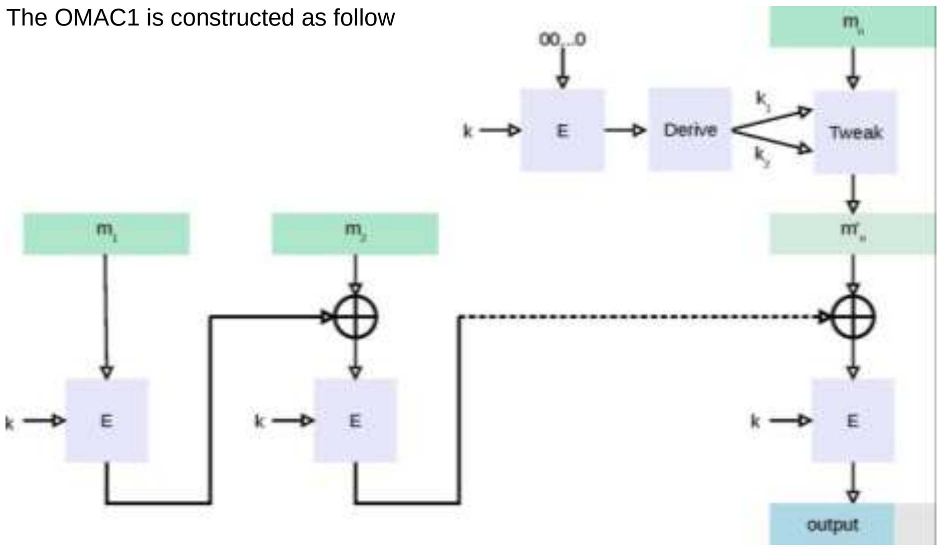
3. Explain the CBC-MAC and provide information how CMAC (OMAC1) addresses problems of the CBC-MAC.

CBC-MAC is based on the CBC block mode of encryption. Basically the ultimate ciphertext is used as the tag.



There are several issues with the CBC-MAC
- The system is broken if you use the same key for encryption and MAC
- The system is also broken for variable length messages as attackers may extend messages with observed tags
- In contrary to encryption the IV needs to be fixed for the MAC and must not be transmitted as this would allow an attacker to flip bits on the ciphertext and try to forge a valid tag by altering the submitted IV.

The OMAC1 is constructed as follow



The change in OMAC1 is basically affecting the last round of chaining where keys are automatically derived (k1, k2) and thus prevent to choose wrong keys. Additionally, the tweak takes care of the message m_n padding and does encrypt m_n. Considering the variable length issue of the CBC-MAC, the encryption of mn prevents that attackers can XOR the tag with m_n using observed tags. Moreover, OMAC1 does not allow for an IV and d

References
- https://en.wikipedia.org/wiki/CBC-MAC
- https://en.wikipedia.org/wiki/One-key_MAC

4. In order to encrypt and integrity protect messages in the same go, Authenticated Encryption (with Associated Data) could be used. You did already study GCM and EAX block modes that provide such features. Please explain the difference of AE and AEAD in simple words.

Terms
- Authenticated encryption (AE)
- Authenticated encryption with associated data (AEAD)

For example, the block modes GCM and EAX can assure confidentiality and integrity in the same go. This is a significant performance advantage over separate processes for encryption and mac-ing.

Assuming you have mixed data of encrypted and unencrypted parts. Using AEAD you may bind the unencrypted part as associated data to the encrypted part and assure integrity for both. E.g for an IP packet carrying encrypted data you could bind the ciphertext to some IP packet information (which is in plaintext of course).

References
- https://en.wikipedia.org/wiki/Authenticated_encryption