

EDUCATION AND TRAINING

CyberSecPro Training

We are creating cutting-edge education and training to advance competencies and professionalism in EU cybersecurity.

OUR VISION

Next level cybersecurity education and training

Cybersecurity Essentials and Management for Energy Sector

CSP001_C_E

PRESENTATION BY:
DAVIDE FERRARIS
UNIVERSITY OF MALAGA, SPAIN

EDUCATION AND TRAINING

CyberSecPro Training

We are creating cutting-edge education and training to advance competencies and professionalism in EU cybersecurity.

OUR VISION

Next level cybersecurity education and training



Co-funded by
the European Union

Acknowledgement

- *Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or HADEA. Neither the European Union nor the granting authority can be held responsible for them.*
- *Project Agreement no. 101083594*

Topic-6: Security Controls Selection and Implementation for Energy Environments

Overview

- Select and implement appropriate security controls based on the specific needs of energy systems
- Implement strong password policies and multi-factor authentication (MFA) to protect user accounts
- Encrypt sensitive data at rest and in transit to prevent unauthorised access and data breaches
- Regularly apply security updates and patches to software systems to address vulnerabilities

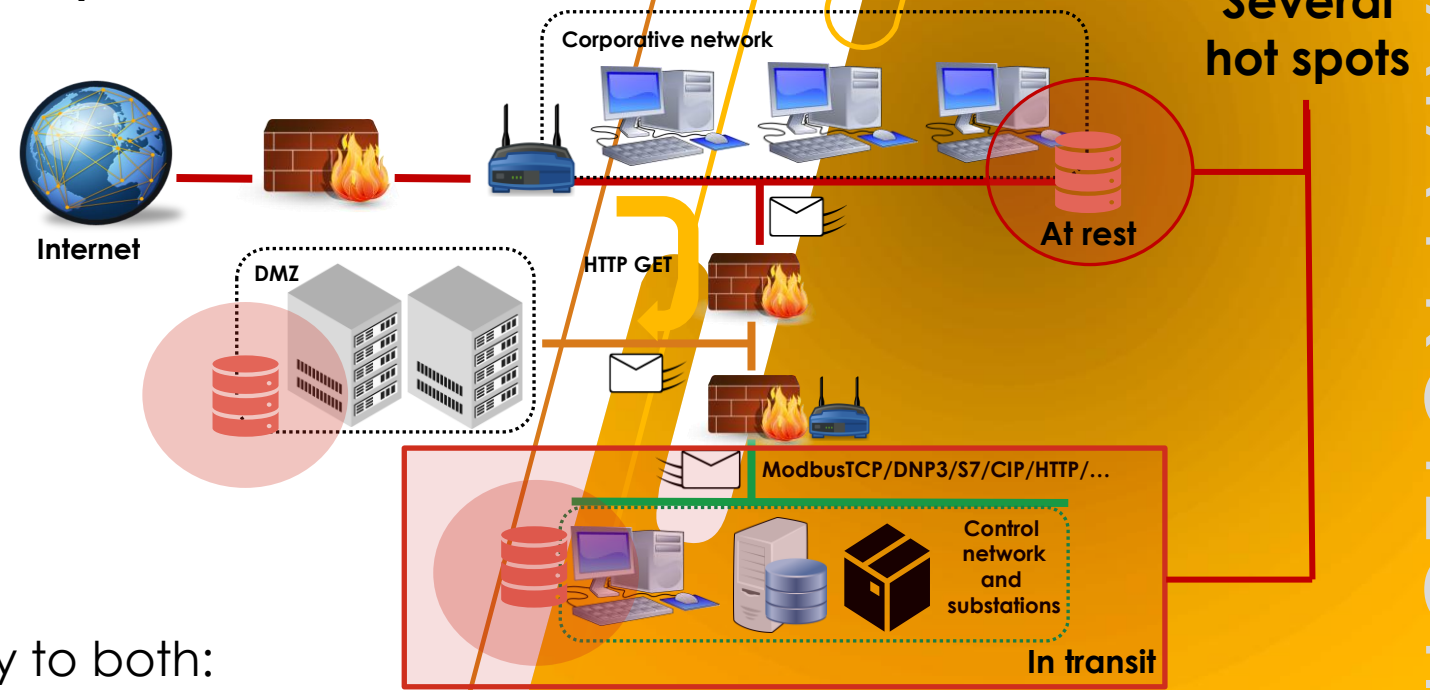
Topic-6: Security Controls Selection and Implementation for Energy Environments

Overview

- Select and implement appropriate security controls based on the specific needs of energy systems
- Implement strong password policies and multi-factor authentication (MFA) to protect user accounts
- **Encrypt sensitive data at rest and in transit to prevent unauthorised access and data breaches**
- Regularly apply security updates and patches to software systems to address vulnerabilities

Cryptography in power control systems

- **Cryptography is a necessary protection measure** in energy control networks
 - Through cryptography, we may prevent unauthorised access to sensitive data, guaranteeing **confidentiality**
- These data can be:
 - Command and control instructions
 - Sensor measurements
 - Network data
 - User requests
 - Authentication data
 - Authorization data
 - Key exchange
 - Etc.
- Moreover, this protection **MUST** apply to both:
 - **Data in transit and data at rest**



Cryptography applied to data in transit

- **Communication channels** of energy control networks **MUST** be protected by applying cryptographic algorithms
 - In charge of protecting unauthorised access to data in transit



Note that in Topic 5 we have already seen that all VPN protocols rely on cryptographic algorithms to guarantee the confidentiality of the communication channels -- which may be based on insecure industrial protocols such as ModbusTCP

- Otherwise, communications would be sent in 'plaintext', allowing attackers with the necessary resources to intercept, capture and read communications (eavesdrop)
- Unfortunately, **eavesdropping** can arise in industrial control networks, as most legacy operating devices still rely on insecure protocols such as **telnet** for the remote control

Cryptography applied to data in transit

- Netresec offers **4SICS Geek Lounge**, offering some files with captures of industrial traffic
 - URL: <https://www.netresec.com/?page=PCAP4SICS>
 - The captures are based on an Industrial Control System (ICS) Lab with PLCs, RTUs, servers and industrial network equipment - *all the information about the ICS Lab is found at the same website*
 - As requested, special acknowledgement goes to CS3Sthlm for allowing the community access to these captures
 - More SCADA/ICS network captures can also be found at: <https://www.netresec.com/?page=PcapFiles>
- With **Wireshark**, it is possible to analyse telnet traffic filtering
 - So an interesting task would be to download the screenshots and filter them by "telnet"

The screenshot shows the Netresec website with the following content:

NETRESEC Experts in network security monitoring and network forensics

NETRESEC | Products | Training | Resources | Blog | About Netresec

NETRESEC » Resources » PCAP Files » 4SICS

Capture files from 4SICS Geek Lounge

The industrial cyber security conference 4SICS (now [CS3Sthlm](#)) is an annual summit that gather the most important ICS/SCADA cyber security stakeholders across critical industries (i.e. energy, oil & gas, water, transportation and smartgrid etc).

The "Geek Lounge" at 4SICS contains an ICS lab with PLCs, RTUs, servers, industrial network equipment (switches, firewalls, etc). These devices are available for hands-on "testing" by 4SICS attendees.

We at Netresec have worked with the 4SICS crew to capture the network traffic at the ICS lab. The idea is to share the captured PCAP files on the Internet, since network traffic from industrial networks is a really scarce resource!

We kindly ask you to credit [CS3Sthlm](#), for allowing the network traffic from the 4SICS lab to be captured and shared, if you will re-distribute this dataset or use it as part of a training. We would also appreciate a link back to this page.

4SICS 2015 PCAP Files

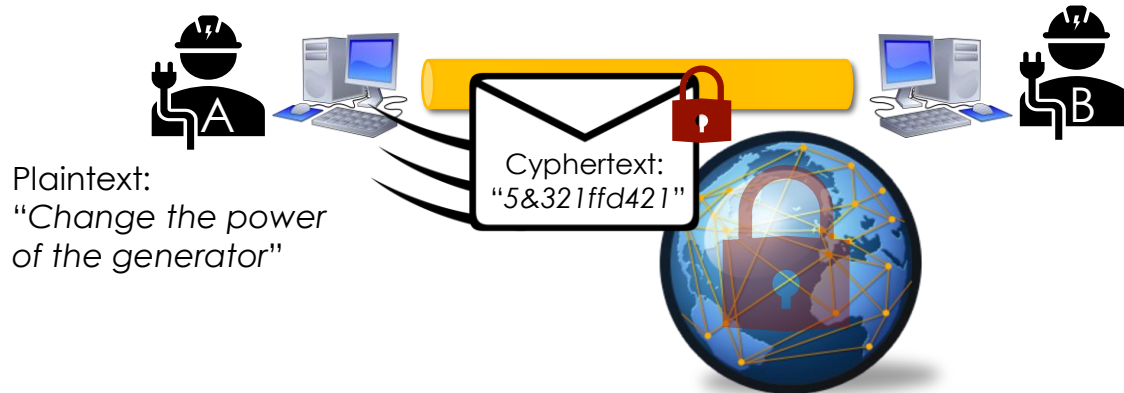
4SICS-GeekLounge-151020.pcap	25MB
4SICS-GeekLounge-151021.pcap	134MB
4SICS-GeekLounge-151022.pcap	200MB

Source and figure source: Netresec, "Captures files from 4SICS Geek Lounge", 2024
 URL: <https://www.netresec.com/?page=PCAP4SICS>
 Source: CS3Sthlm, 2014-2020, accessed in 2024.
 URL: <https://cs3sthlm.se>



Cryptography applied to data in transit

- There is therefore a **clear need to protect communication** channels in energy control systems, where multiple industrial equipment must coexist for control



- This type of protection involves:
 - An **encryption** phase, in which the data is protected by combining the 'plaintext' with a 'secret', resulting in a 'ciphertext'
 - A **decryption** phase, where the 'ciphertext' is transformed into the original 'plaintext' by means of a secret
- This **secret** = the "KEY"



Cryptography: Types

- Within modern cryptography, three relevant cryptographic areas have emerged

Symmetric



- Symmetric cryptographic algorithms use the **same key** for both encryption and decryption

Asymmetric / public key



- Asymmetric cryptographic algorithms use **two different keys** to carry out the encryption and decryption process

Hybrid



- The hybrid technique aims to **combine the asymmetric and symmetric algorithms**

Cryptography: Types

- Within modern cryptography, three relevant cryptographic areas have emerged

Symmetric



- Symmetric cryptographic algorithms use the **same key** for both encryption and decryption

Asymmetric / public key



- Asymmetric cryptographic algorithms use **two different keys** to carry out the encryption and decryption process

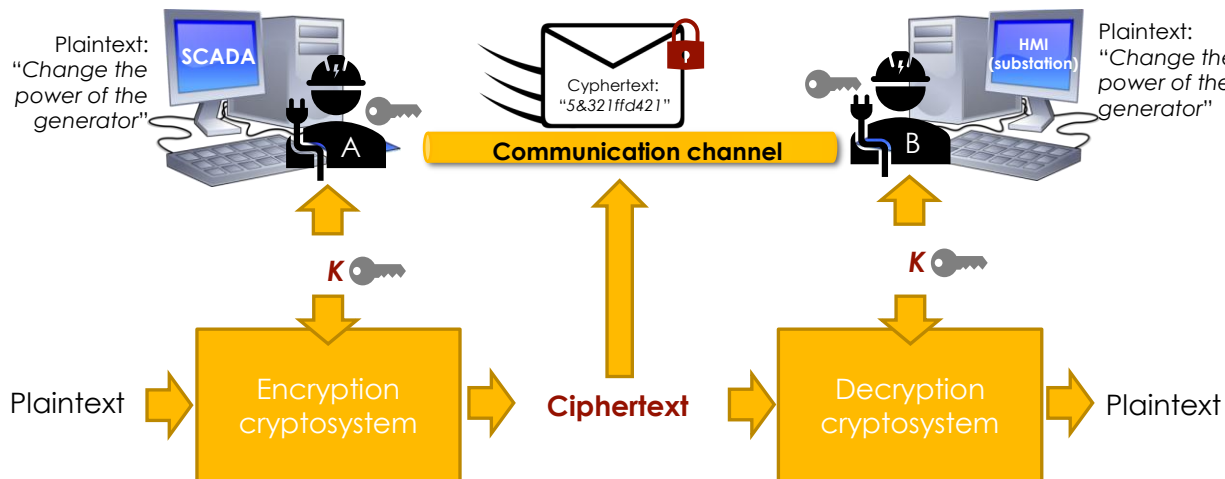
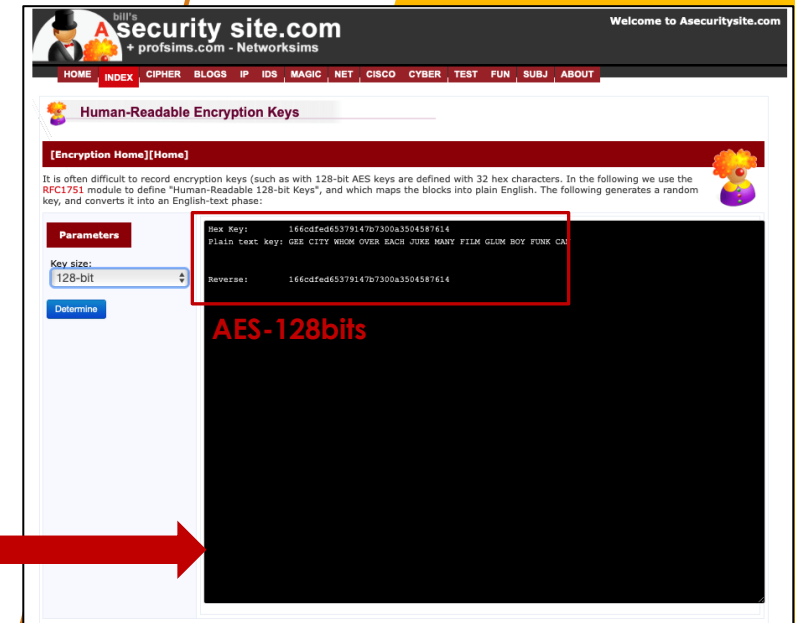
Hybrid



- The hybrid technique aims to **combine the asymmetric and symmetric algorithms**

Symmetric cryptography: Procedure

- Basically, the technique aims to:
 1. IT/OT administrators, industrial network equipment or processes MUST first agree the encryption conditions such as:
 - Type of encryption/decryption algorithm
 - The 'session' key, such that one of the two entities generates the key using some a (pseudo-random) generator
 2. One of the entities involved, encrypt the message under the pre-set conditions and send the encrypted message



Source and figure source: Buchanan, William J., Human-Readable Encryption Keys. Asecuritysite.com, 2024.
 URL: <https://asecuritysite.com/encryption/plain>

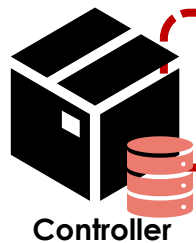
Symmetric cryptography: Cryptosystems

Features	DES	3DES	AES	Camellia
Name	Data Encryption Standard	Triple DES	<i>Advanced Encryption Standard</i>	Camellia
Author	IBM	IBM	<i>Vincent Rijmen, Joan Daemen</i>	Mitsubishi Electric y NTT
Key length	56 bits	168 (3 claves) o 112 (2 claves)	<i>128, 192, 256 bits</i>	128, 192, 256 bits
Cipher block size	64 bits	64 bits	<i>128 bits</i>	128 bits
Processing speed	Slow	Very slow	<i>Fast</i>	Slow
Security	Low – It is not recommended	Medium	<i>High</i>	High

Both are the most widespread and recommended cryptosystems in transit and at rest - both are part of TLS-v1.2/1.3

Symmetric cryptography: Cryptosystems

Features	DES	3DES	AES	Camellia
Name	Data Encryption Standard	Triple DES	<i>Advanced Encryption Standard</i>	Camellia
Author	IBM	IBM	<i>Vincent Rijmen, Joan Daemen</i>	Mitsubishi Electric y NTT
Key length	56 bits	168 (3 claves) o 112 (2 claves)	<i>128, 192, 256 bits</i>	128, 192, 256 bits
Cipher block size	64 bits	64 bits	<i>128 bits</i>	128 bits
Processing speed	Slow	Very slow	<i>Fast</i>	Slow
Security	Low – It is not recommended	Medium	<i>High</i>	High



However, we cannot forget that there are still legacy operational devices that apply these cryptosystems

Symmetric cryptography: A practical view

- Through the online tool **cryptii** (<https://cryptii.com>), we can put the lessons learned into practice
 - Got to 'Modern Cryptography' > Block Cipher
- Exercises:
 1. Encrypt the following message 'Welcome to this module!', using the following Key: `2b7e151628aed2a6abf7158809cf4f3c`
 2. Decrypt the ciphertext using the same key and encryption conditions
- What is happening?
 - Wuala ! 😊
 - We can encrypt messages and recover the original message

The screenshot displays the Cryptii web application interface, which is divided into four main sections from left to right:

- Text Input:** A text area containing the message "Welcome to this module!".
- Block Cipher Configuration:** A panel where the encryption settings are defined. The algorithm is set to "AES-128", the mode to "CBC (Cipher Block Chaining)", and the key to "2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c". The IV is set to "00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f". The output is labeled "Encoded 32 bytes".
- Block Cipher Configuration:** A second panel with identical settings to the first, but the output is labeled "Decoded 23 bytes".
- Text Output:** A text area showing the result of the decryption, which is "Welcome to this module!".

Symmetric cryptography: A practical view

- We can enhance our technical knowledge and extend our 'SKILLS' by considering the online tool of **Asecuritysite**
 - AES: <https://asecuritysite.com/symmetric/aes>
 - DES: <https://asecuritysite.com/symmetric/des>
 - 3DES: <https://asecuritysite.com/symmetric/threedes>
 - Camellia: <https://asecuritysite.com/symmetric/camellia>
- Exercises:
 1. Obtain the session key, where the seed (a special pre-key) for the generation of the key could be "1234"
 2. Encrypt and decrypt a message, e.g., "Hello World !"
 3. If you have time, take a look at the attached code on the same web page

The screenshot displays the Asecuritysite.com interface for an AES example. It includes a navigation bar, a header with the site name, and a main content area. The 'Text and key' section contains a form where a message 'Hello world !' and a key '1234' are entered. The 'Ciphertext' section shows the encrypted message in Base-64 and Hex. The 'Plaintext' section shows the decrypted message 'Hello world !'. The 'Code Used' section contains a C# code snippet for AES encryption and decryption.

Source and figure source: Buchanan, William J., *AES*. Asecuritysite.com, 2024.

URL: <https://asecuritysite.com/symmetric/aes>

Source: Buchanan, William J., *DES Cipher*. Asecuritysite.com, 2024.

URL: <https://asecuritysite.com/symmetric/des>

Source: Buchanan, William J., *3DES Cipher*. Asecuritysite.com, 2024.

URL: <https://asecuritysite.com/symmetric/threedes>

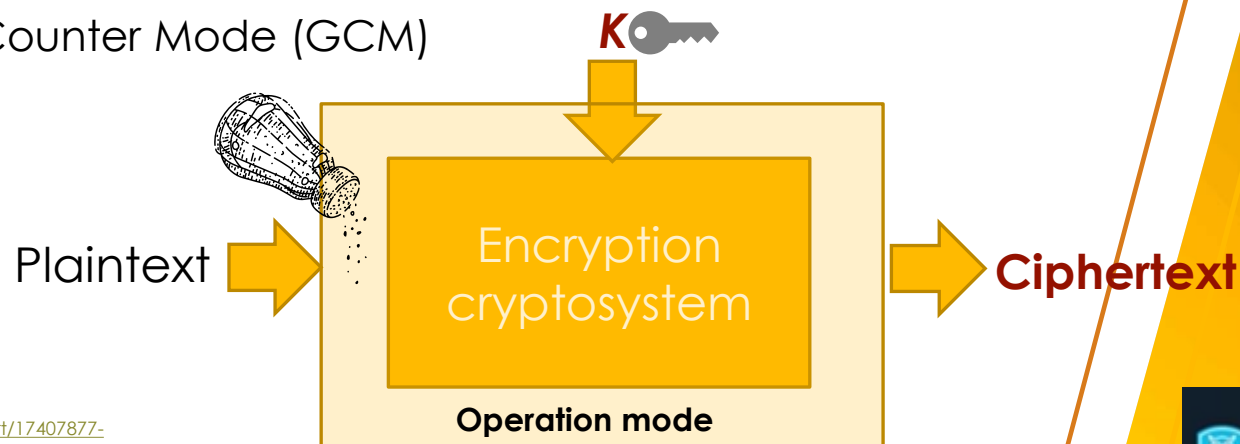
Source: Buchanan, William J., *Camellia cipher*. Asecuritysite.com, 2024.

URL: <https://asecuritysite.com/symmetric/camellia>



Symmetric cryptography: 'Modes of operation'

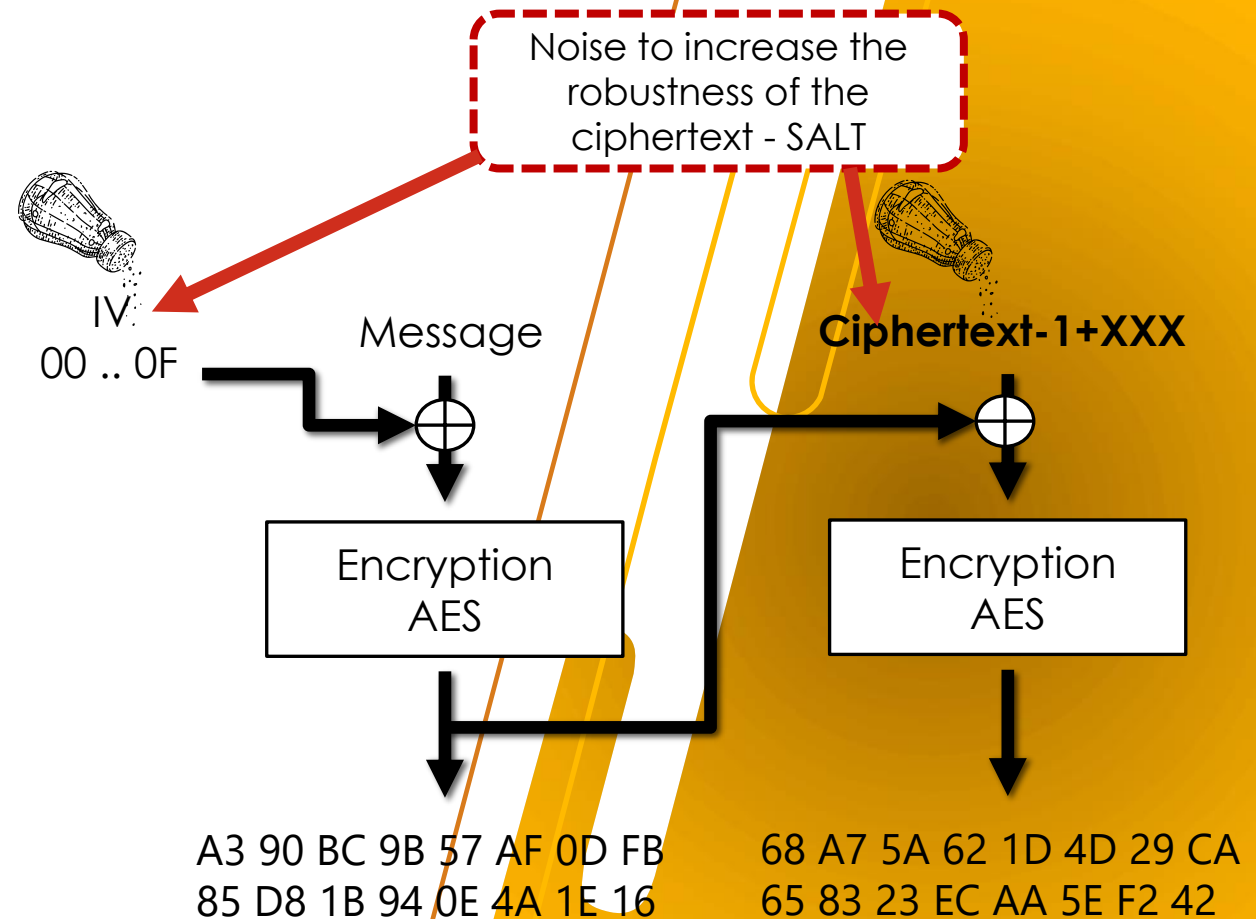
- Note that if the symmetric cryptographic algorithms were used as they are, it would be possible to obtain the same ciphertext every time
 - That is, for the same plaintext as input, the same ciphertext will always be obtained as output !
- To avoid this situation, block cipher **modes of operation** are applied to **inject noise** (SALT) into the encryption process:
 - Cipher Block Chaining (CBC)
 - Counter (CTR)
 - Galois/Counter Mode (GCM)



Symmetric cryptography: 'Modes of operation'

• CBC mode

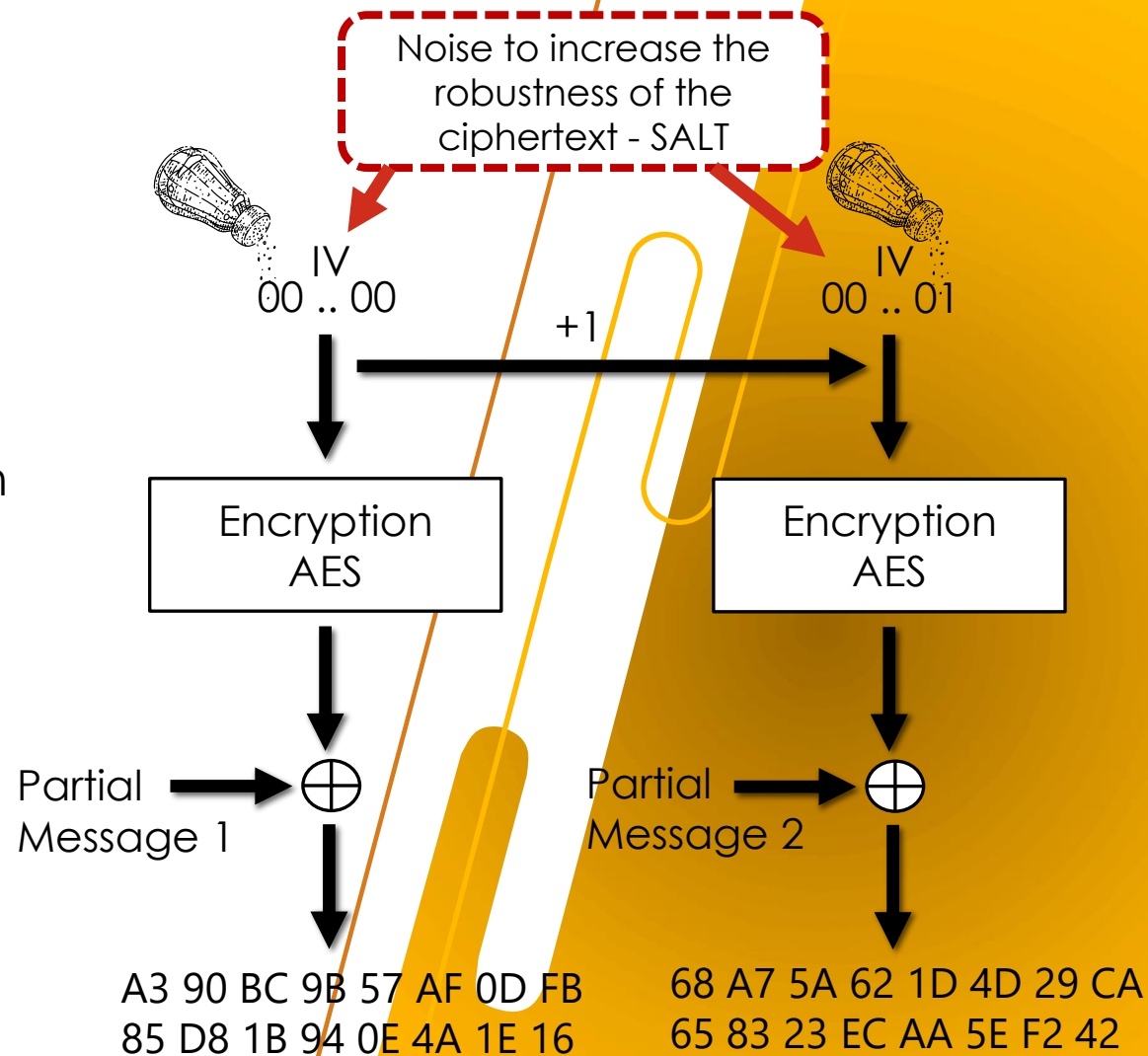
- The output of each cipher block is used to encrypt the next block
- Uses a random '**Initialisation Vector**' (IV) as input to the first cipher - *provides variability (= SALT)*
 - Note that this IV value is public (e.g. 00..0F) and must be sent the destination node together with the ciphertext
- A padding operation is normally performed at the end of the process to facilitate the computation
 - 'Pad the input (XXX) to be a multiple of the block size



Symmetric cryptography: 'Modes of operation'

- **CTR mode**

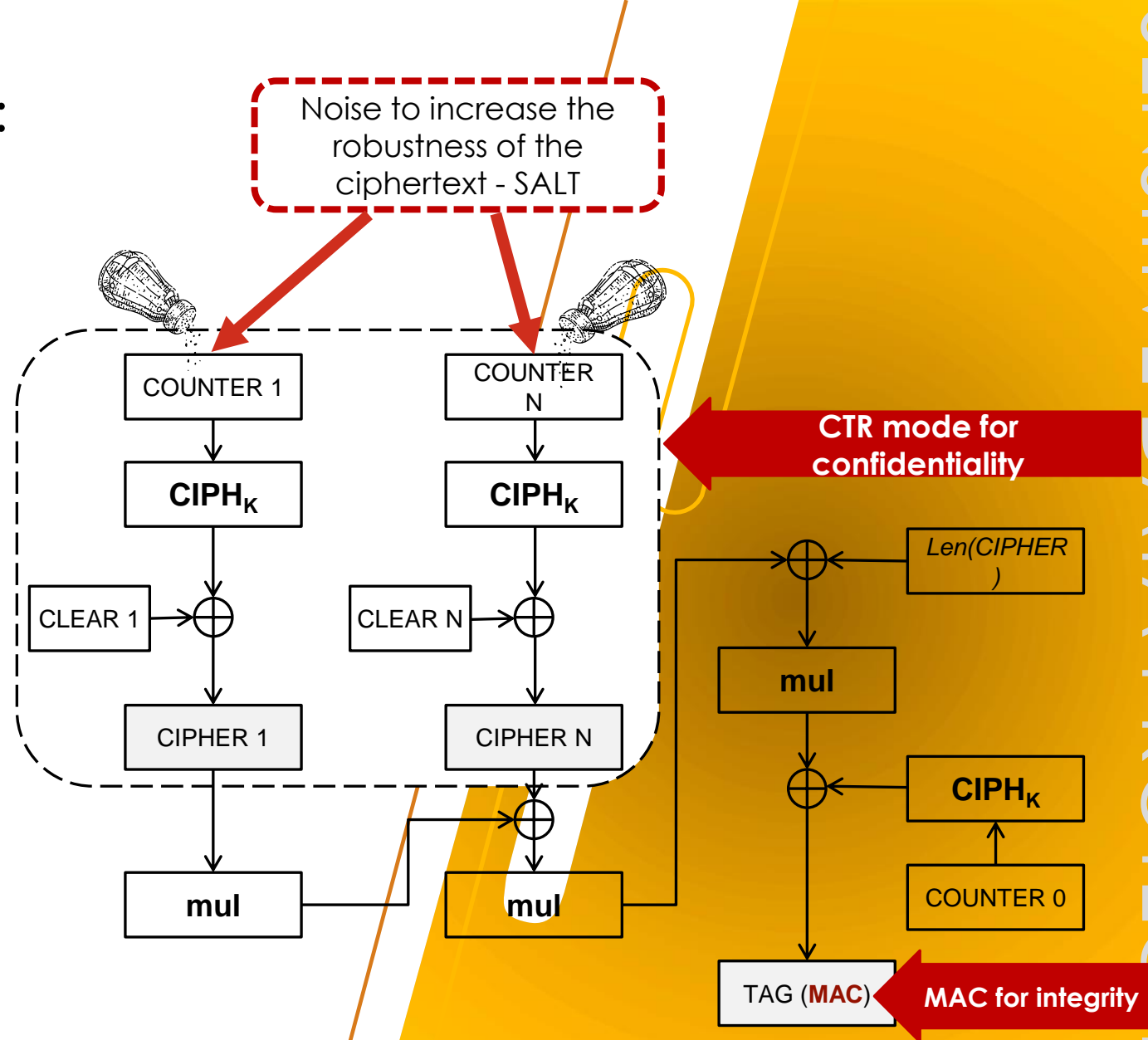
- The result of each cipher block is part of the ciphertext
- When all operations are completed, each ciphertext piece is concatenated to provide the final ciphertext piece
- This operation model also uses a random '**Initialisation Vector**' (IV) as input for the first cipher
 - Its value is increased for subsequent operations (= SALT)



Symmetric cryptography: 'Modes of operation'

• GCM mode

- Uses a random value ("nonce" / "counter" / IV) as input to the first cipher - *provides variability (= SALT)*
- This value is public and must be sent along with the ciphertext
- No padding operation to be done
- Provides a cipher integrity code ("Tag" / "MAC")
 - A change in the cipher changes the integrity code
 - This features comply with the **Authenticated Encryption with Additional Data (AEAD)** property – *confidentiality + integrity*



Modes of operation: A practical view

- Go again to **cryptii** to 'Modern Cryptography' > Block Cipher
- Exercises:
 1. Encrypt the following message 'Welcome to this module!', using:
 - Key: 2b7e151628aed2a6abf7158809cf4f3c
 - Operation mode: CTR
 - IV: 000102030405060708090a0b0c0d0e0f
 2. Decrypt the ciphertext but using the same encryption conditions

The screenshot displays the Cryptii web application interface, which is divided into four main sections from left to right:

- Text Input:** A text area containing the message "Welcome to this module!".
- Block Cipher Configuration (Left):** A panel with tabs for "ENCODE" and "DECODE". It shows the following settings:
 - ALGORITHM: AES-128
 - MODE: CTR (Counter)
 - KEY: 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
 - IV: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 - Output: → Encoded 23 bytes
- Block Cipher Configuration (Right):** A panel with tabs for "ENCODE" and "DECODE". It shows the same settings as the left panel:
 - ALGORITHM: AES-128
 - MODE: CTR (Counter)
 - KEY: 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
 - IV: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 - Output: → Decoded 23 bytes
- Text Output:** A text area containing the message "Welcome to this module!".

Modes of Operation: A practical view

- Go again to **Asecuritysite** and particularly to:
 - URL: <https://asecuritysite.com/symmetric/sym>
- Exercises:
 1. Encrypt with AES+CBC, AES+CTR and AES+GCM
 2. Decrypt the ciphertext but using the same encryption conditions

```
Type: AES
Mode: CTR
Message: Hello world !
Message with padding: b'Hello world !\x03\x03\x03'

Key: 8b2e8c79cae966f0ebcec32bf9635aab
IV: 45dea0e10df27d2138305bc8bf69cc9e

Cipher: eb44aa5019cfcfc477ec47847c115db5
Decrypt: Hello world ! Result
```

```
Type: AES
Mode: CBC
Message: Hello world !
Message with padding: b'Hello world !\x03\x03\x03'

Key: c6169ddc47170a0927d6887833356406
IV: 9ef0641645be0e5a0fe92c9a6295fa68

Cipher: 2dab11ae0b305ae33cf9b69af7e7d91
Decrypt: Hello world ! Result
```



```
Type: AES
Mode: GCM
Message: Hello world !
Message with padding: b'Hello world !'

Key: 4d0829231dcd7d675b3a564dedbef604
IV: 827a552a913288c12345669fecb3e786

Cipher: 12636538fab8f06cd23c384248
Tag: 66f2b0699bcbca92c23e1460ed53ea9d
Decrypt: Hello world ! Result
```

MAC for integrity

Symmetric cryptography: Advantages and disadvantages

Advantages	Disadvantages
<ul style="list-style-type: none"> Symmetric encryption algorithms typically offer high performance at low (or relatively low) computational cost 	<ul style="list-style-type: none"> The secret key <i>K</i> has to be agreed a priori 
<ul style="list-style-type: none"> The keys used are relatively short It is recommended to use keys of 128-bits size or larger 	<ul style="list-style-type: none"> The number of keys may grow significantly if all users of a community need to communicate (individually) with the other members of that community $\rightarrow (n * (n-1)) / 2$

Cryptography: Types

- Within modern cryptography, three relevant cryptographic areas have emerged

Symmetric



- Symmetric cryptographic algorithms use the **same key** for both encryption and decryption

Asymmetric / public key



- Asymmetric cryptographic algorithms use **two different keys** to carry out the encryption and decryption process

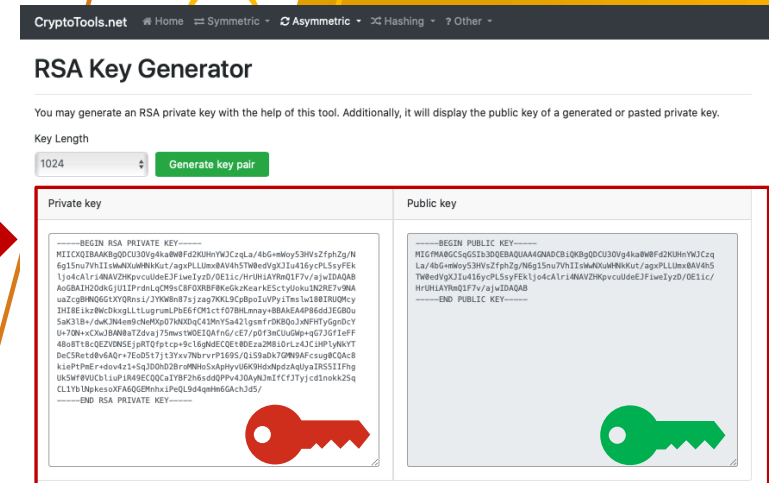
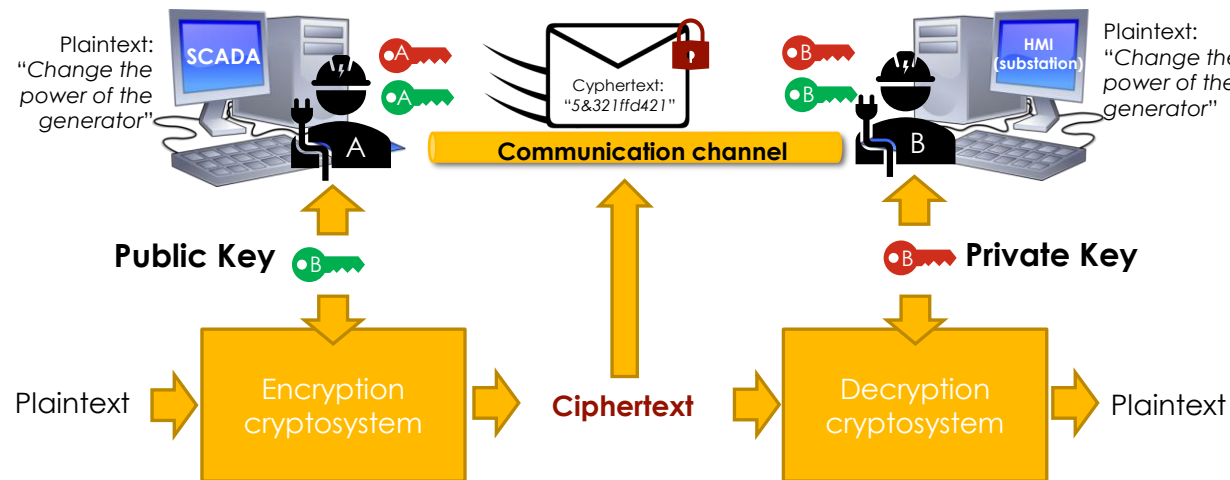
Hybrid



- The hybrid technique aims to **combine the asymmetric and symmetric algorithms**

Asymmetric cryptography: Procedure

- Basically, the technique aims to:
 1. Each IT/OT administrator, industrial network equipment or process **MUST** create two relevant keys:
 - **1 Private Key**: known by only the owner of the key
 - **1 Public key**: known by all entities
 2. One of the entities involved **encrypts** the message using the 'public key' of the destination as it is the only entity with the private key

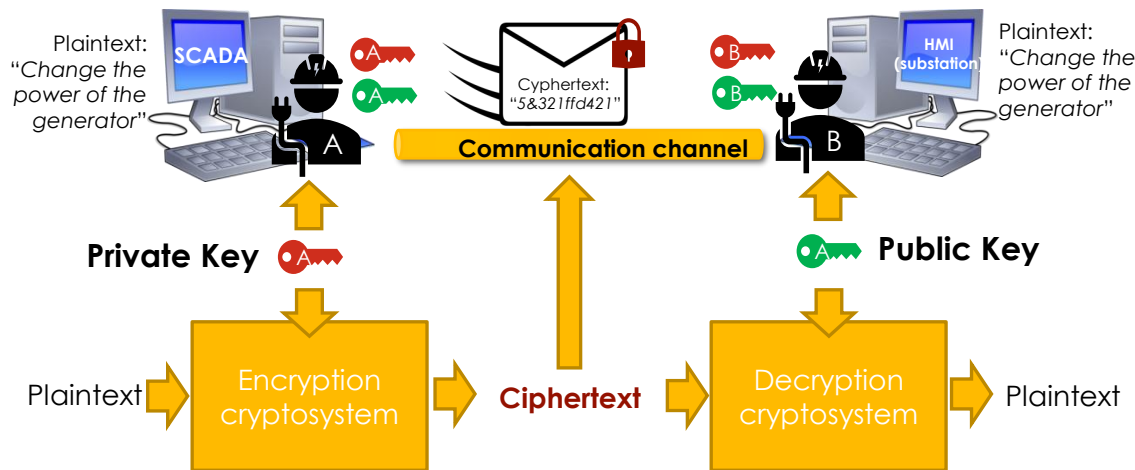


These two keys are different but closely related to each other, allowing the encryption and decryption of information

Source and figure source: CryptoTool.net, RSA Key Generator, 2024.
 URL: <https://cryptotools.net/rsagen>

Asymmetric cryptography: Procedure

- However, if the private key is applied to encrypt (sign) the message, what is achieved is a **Digital Signature**



- Thus, depending on how the two keys are applied, this is achieved:

Encryption for confidentiality	Digital Signature for authentication
<ul style="list-style-type: none"> • Encrypt with the receiver's public key • Decrypt with the receiver's private key 	<ul style="list-style-type: none"> • Encrypt with the sender's private key • Decrypt with the sender's public key

Asymmetric cryptography: Cryptosystems

Features	RSA	ECC
Name	RSA	Elliptic-curve cryptography
Author	Ron Rivest, Adi Shamir and Leonard Adleman	Neal Koblitz and Victor S. Miller
Key length	1024, 2048 y 3072 bits	160, 224, 256, 384, 521 bits
Processing speed	Slow	Fast
Security	High	High
Functionality	Encryption, signature and key exchange	Encryption, signature and key exchange

RSA is the most widespread, but ECC is ideal for computationally constrained operational devices, such as RTUs/PLCs, sensors and actuators

Asymmetric cryptography: Advantages and disadvantages

Advantages	Disadvantages
<ul style="list-style-type: none">All entities involved in the energy control system can generate the two keys without having to agree on them a priori	<ul style="list-style-type: none">Asymmetric encryption algorithms require computational capabilities to process their encryption algorithms
<ul style="list-style-type: none">El number of keys is $2*n$Reduces the cost of storage at end points	<ul style="list-style-type: none">The keys they use are relatively largeIt is recommended to use keys of size 1024 bits or larger

Cryptography: Types

- Within modern cryptography, three relevant cryptographic areas have emerged

Symmetric



- Symmetric cryptographic algorithms use the **same key** for both encryption and decryption

Asymmetric / public key



- Asymmetric cryptographic algorithms use **two different keys** to carry out the encryption and decryption process

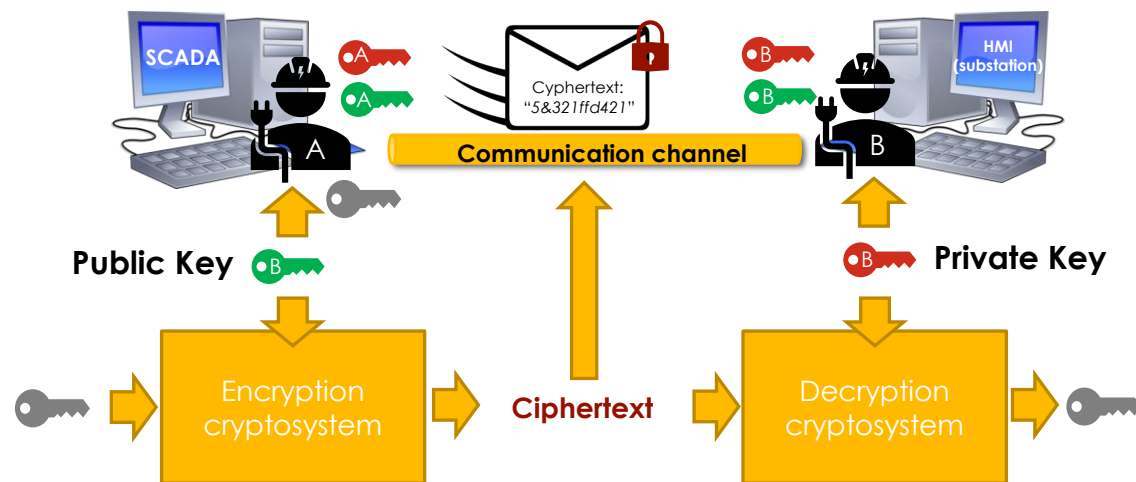
Hybrid



- The hybrid technique aims to **combine the asymmetric and symmetric algorithms**

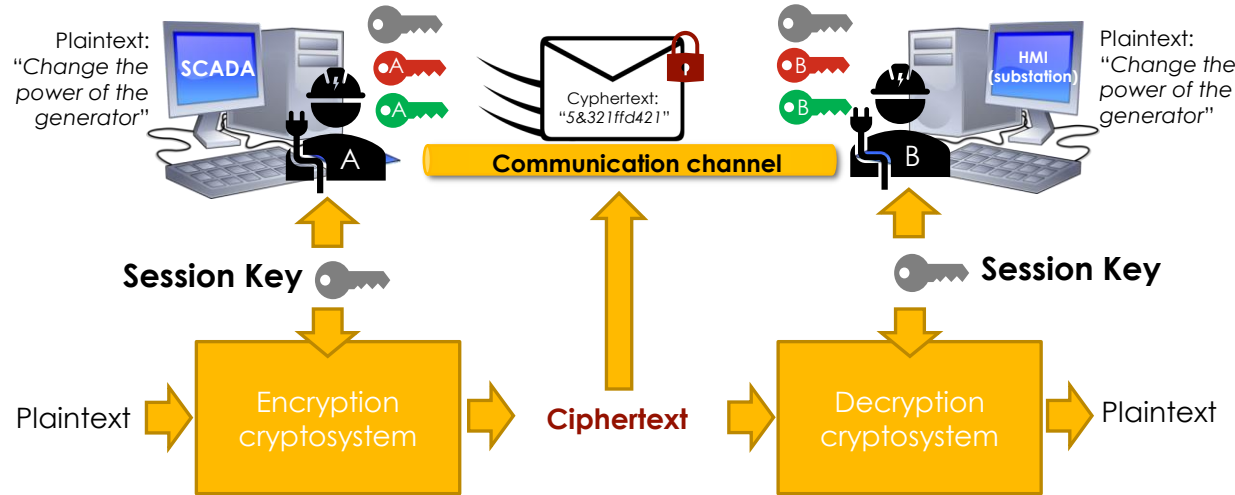
Hybrid cryptography: Procedure

- Hybrid cryptography aims to obtain the advantages offered by the two approaches:
 - **Asymmetric cryptography** to negotiate the session key
 - **Symmetric cryptography** to encrypt/decrypt messages
- Basically, the technique aims to:
 1. One of the entities generates the session key, and sends it to the receiver using asymmetric cryptography



Hybrid cryptography: Procedure

- Once the session key has been obtained by the receiver, one of the entities generates a message, which is protected using the session key



- However, the exchange of public keys without additional information about the entity, only the key (hexadecimal values), brings with it the need to manage **digital certificates** in order to trust the public key

B Public Key

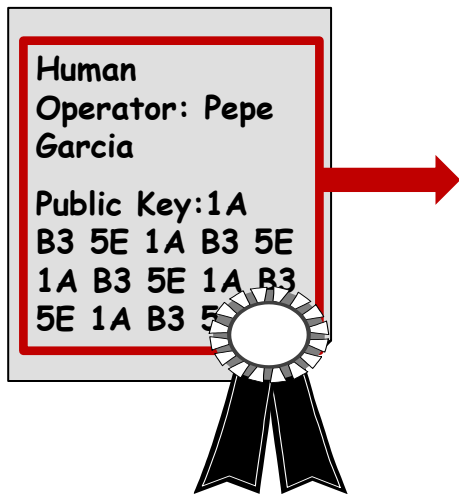
```

98 3f ad 19 36 93 3d 3e fe 47
fa ad 22 7a 58 e3 46 d0 5d c1 2d 8f 31
5e fe b4 30 fe 50 74 ac d6 9d 82 c6 49 dd
14 12 7d 71 0b ac 06 c1 3f d7 06 87 e0 90 89 d6
e5 e3 03 b2 f2 27 b1 9f 33 c8 aa 6b 36 4a a3 c4
3f 79 41 9d 89 46 2f 2b 3e 63 d4 38 56 91 aa 1d
b1 0d 42 75 4d f3 87 4e e3 0f 4d cc b4 6c bf 62
13 87 ea d0 9b 8e b6 e2 ff 19 f4 94 09 d5 96 61
    
```

Digital certificates: x.509

- A digital certificate is the digital document with the "seal of assurance" that a key belongs to a specify entity, mainly because it is certified by a **trusted entity** that validates and signs the document
- This entity in charge of validating and certifying is known as a **Certification Authority (CA)**

The CA should be the entity with legal rights to certify the document, such as the system's managers or directives, etc.

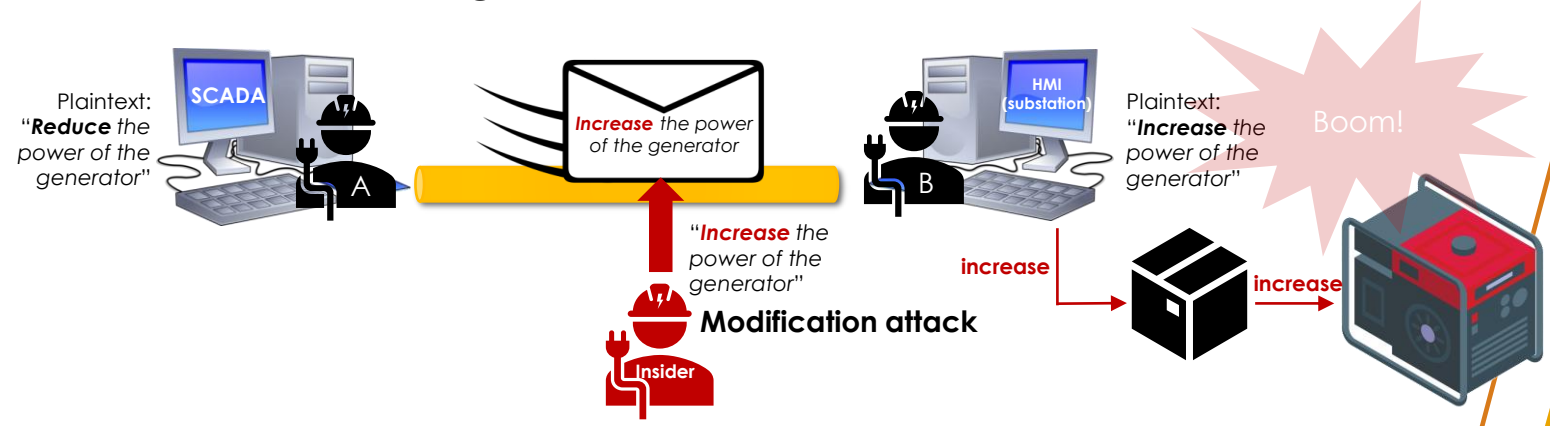


The document contains at least

- A unique serial number
- The identity of the user for whom the information is provided
- The value of the public key
- The expiry date of the certificate
- The identity of the issuer of the document
- The digital signature of the document issue

Hash for sensitive data integrity

- One way to detect unwanted changes to the content of messages would be through specific cryptographic functions for "integrity"



- **Hash functions** are the most cryptographically efficient validation methods
 - These functions are **one-way functions** that ensure that two different messages result in different hash values (or digests)
 - With this, we also ensure "**collision-free**" (the main vulnerability of such functions)

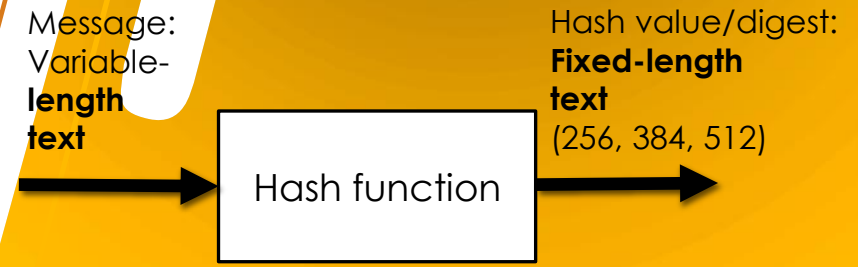
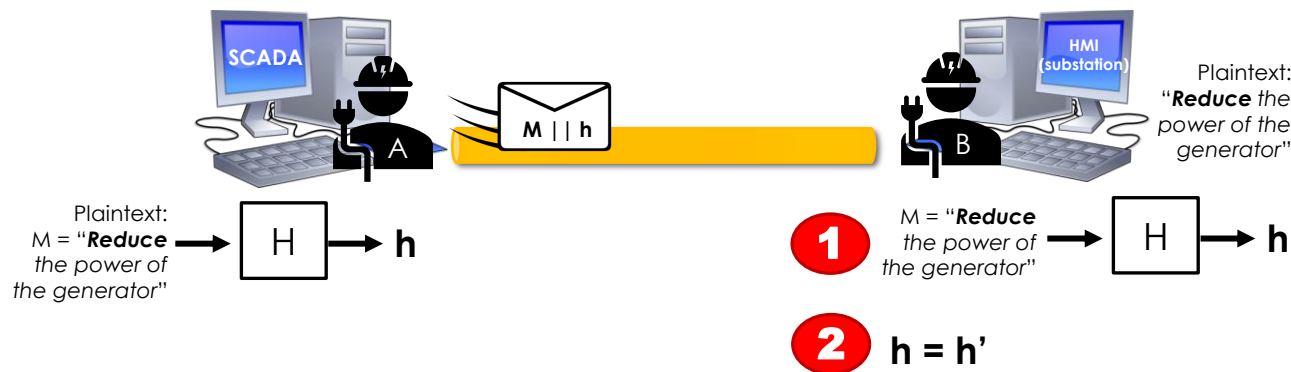


Figure source: Vecteezy
 URL: <https://www.vecteezy.com/vector-art/19053049-petrol-generator-icon-isometric-vector-industrial-equipment>

Hash function: Procedure

- Basically, the technique aims to:
 1. The sender prepares the message and obtain its corresponding hash value
 2. Both the message and the hash value are sent to the destination
 3. The receiver calculates the hash value of the received message and compares it with the received hash value
 4. If both hashes are equal, the receiver confirms the integrity of the message



Hash function: A practical view

- Go to **Asecuritysite**, and particularly to:
 - URL: <https://asecuritysite.com/encryption/md5>
- Exercises:
 - Generate the hash of a message
 - Change a character to the message
 - Compare the hash values
- Repeat the exercise with **criptii**

It is important to note that some hash functions are susceptible to collisions, such as: MD5 and SHA-1

- Unfortunately, **legacy operating devices can still support them**

Source: Buchanan, William J, Hashing, Asecuritysite.com, 2024.
 URL: <https://asecuritysite.com/encryption/md5>
 Source: Wierk, Cryptii, 2024.
 URL: <https://cryptii.com>

Hash Example (Hex)

[Encryption Home][Home]

MDS, SHA-1 and SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512) produce a hash signature, and the output is typically shown in a hex format or a Base-64. In this example the output is converted into a **Hex** format. MD5 and SHA-0 have been shown to have collision, and are prone to attacks. Along with this SHA-1 has been shown to be prone to the theoretical attack. SHA-2 is not susceptible to these attacks.

Message:

Hex input:

The results are then:

MD5	67C18D060479C5D867C9B91C80EDEB4C
SHA-1	7C0A529D2E9E40F54944674B0DE7E806FBA33262
SHA-256	2F951D3ADF29AB254D734286755E2131C397B6FC1894E6FFE5B236EA5E099ECF
SHA-384	B66AC9FBA732BCBD7CAB76CDAA883A5FA482E7028F36B98615F7320106549F66DE2999AAB17E3C
SHA-512	6389D2C21CB35908D355B7DAE876DF2EC9DA2B5920542638BB72DDCB17C9C0A3FF37C1162F5DC

[Encryption Home][Home]

MDS, SHA-1 and SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512) produce a hash signature, and the output is typically shown in a hex format or a Base-64. In this example the output is converted into a **Hex** format. MD5 and SHA-0 have been shown to have collision, and are prone to attacks. Along with this SHA-1 has been shown to be prone to the theoretical attack. SHA-2 is not susceptible to these attacks.

Message:

Hex input:

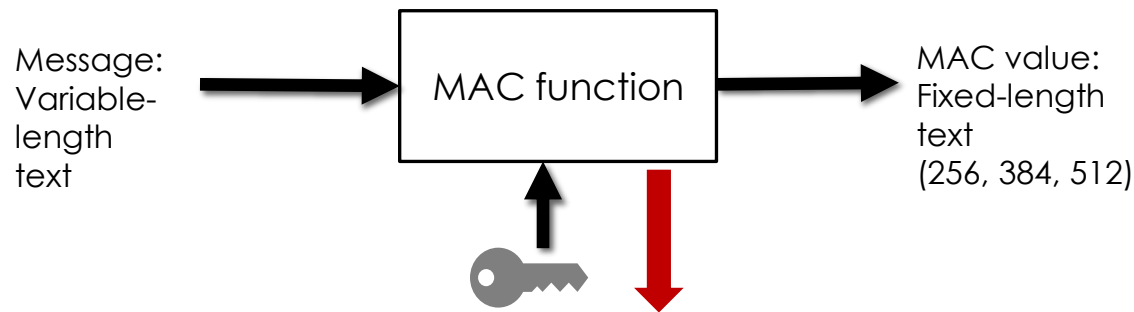
The results are then:

MD5	0587555C4F2316082AD2BF63A1C7E180
SHA-1	31DA2A3B7919EAEF474BC006D066BDC68356E43E
SHA-256	66026CBF32D91644EC10EEC723AC2D54B3B3FCCB07CE6E242080BA727229A775
SHA-384	C2B7FD1C538CE8564C9BFE2E7DDB84143F08A59DA3019FBC2FD1B0C40BEC712AB183EFA245612
SHA-512	DB17E626455AD7E9C85126F5BBFBA0422C795ADF98A151B6B1A1C01A4D7267506FBA3D55689DC

They are completely different hashes from each other

MAC for data integrity and authentication

- The **Message Authentication Code (MAC) function** does the same as the hash function, but considering as input values both:
 - The message, and
 - A symmetric KEY shared by the two entities of the communication



There are several MAC functions, such as **Hash-Based Message Authentication Codes (HMAC)**

- MAC guarantees:
 - **Integrity** through the hash function (e.g. HMAC)
 - **Authentication** through the secret key

As in the previous points, learners can practice with HMAC using the online tools:

- **Asecuritysite**,
https://asecuritysite.com/mac/go_hmac
- **Cryptii**,
<https://cryptii.com>

Cryptography applied to e-mails (data in transit)

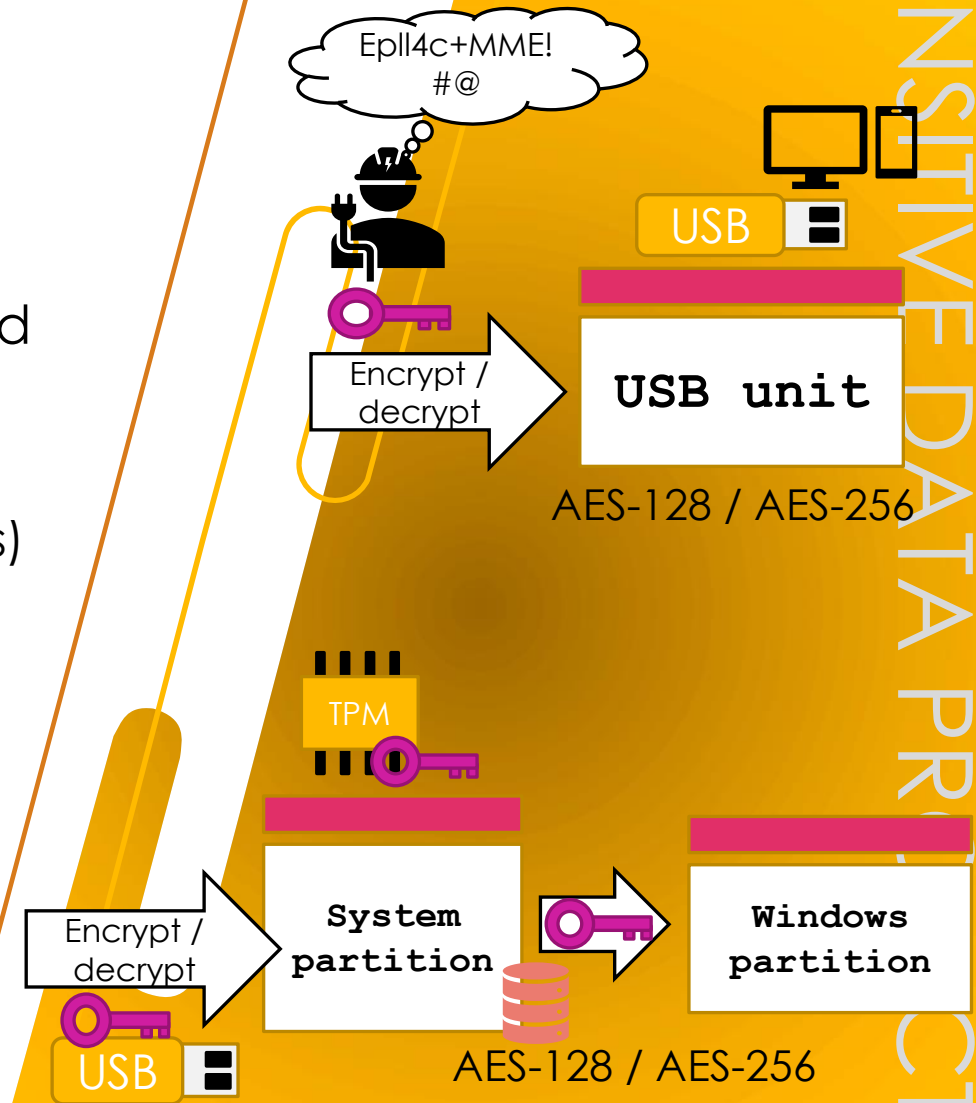
- Energy professionals can activate security measures that allow them to protect their e-mails, using, for example:
 - **Secure/Multipurpose Internet Mail Extension (S/MIME)**
 - **Pretty Good Privacy (PGP)** under the OpenPGP standard
- Both protocols offer **confidentiality, integrity and authentication** guarantees
 - S/MIME handles digital certificates x.509 for authentication
 - PGP offers its own certificate format
- S/MIME and PGP are compatible for all the platforms and can be configured in multiple email editors, such as Thunderbird
 - Also note that PGP can imply the installation of GnuPG, which is an implementation of the OpenPGP standard
 - And PGP can be applied to protect data at rest, such as files

As in the previous points, learners can also practice the potential of PGP to protect files locally, using, for example:

- **Kleopatra**
(open-source, integrating GnuPG)
<https://www.openpgp.org/software/kleopatra/>

Cryptography applied to data at rest

- Operating Systems normally support cryptographic applications to encrypt/decrypt sensitive data on hard disk and removable drive
 - Windows 10/11: **Bitlocker** (by default)
 - Linux and Windows: **Veracrypt, TrueCrypt** (open sources)
- The protection of these units usually follows common procedures:
 - **USB driver protection** relies on a password to obtain the 'key' needed to operate the unit by means of an encryption/decryption algorithm
 - **Hard disk protection** may require a Trusted Platform Module (TPM) 1.2 (or a USB driver) to obtain the key responsible for protecting the disk partition



Veracrypt – an example of its usefulness

- **Open source:**

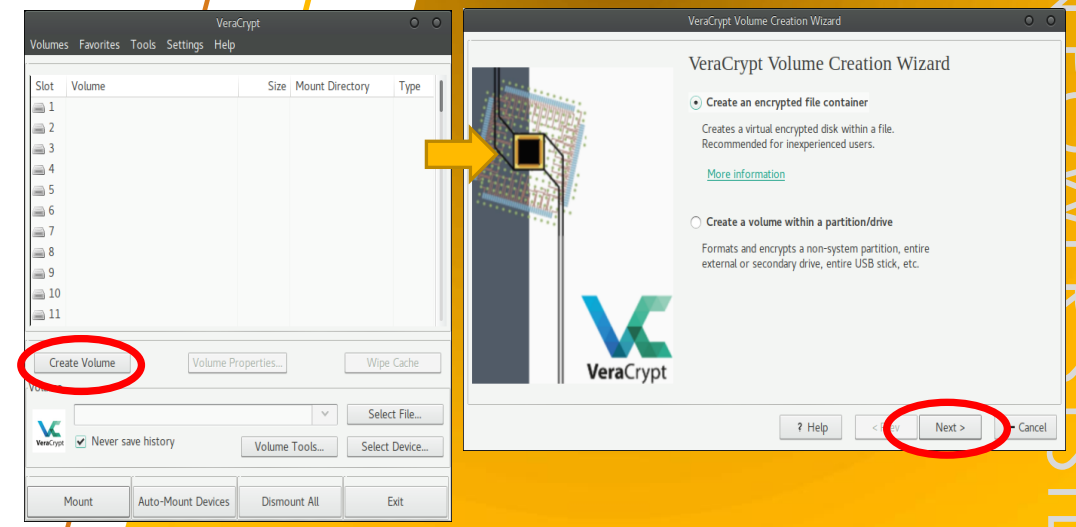
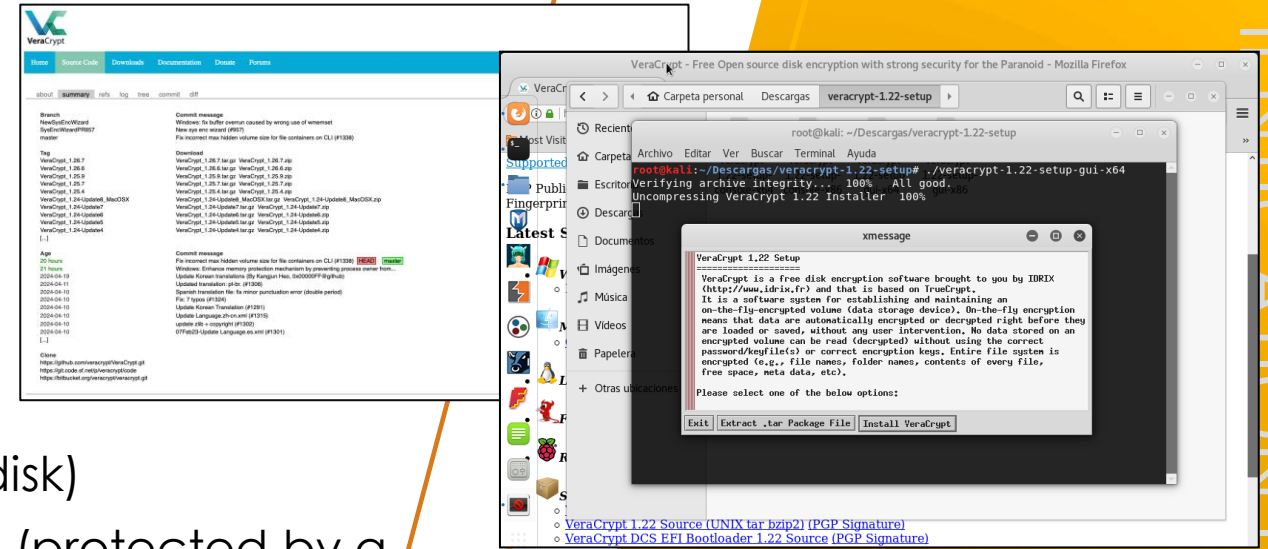
- <https://www.veracrypt.fr/code/VeraCrypt/>

- **Main functions:**

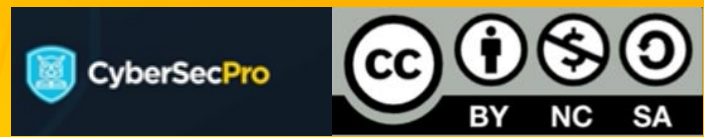
- Encryption of units (USB flash drive or hard disk)
 - Creation of data volumes and hidden units (protected by a password)

- Example of **creating secure data volumes:**

1. Select 'encrypted file container', to store the data inside an encrypted virtual volume in a single file



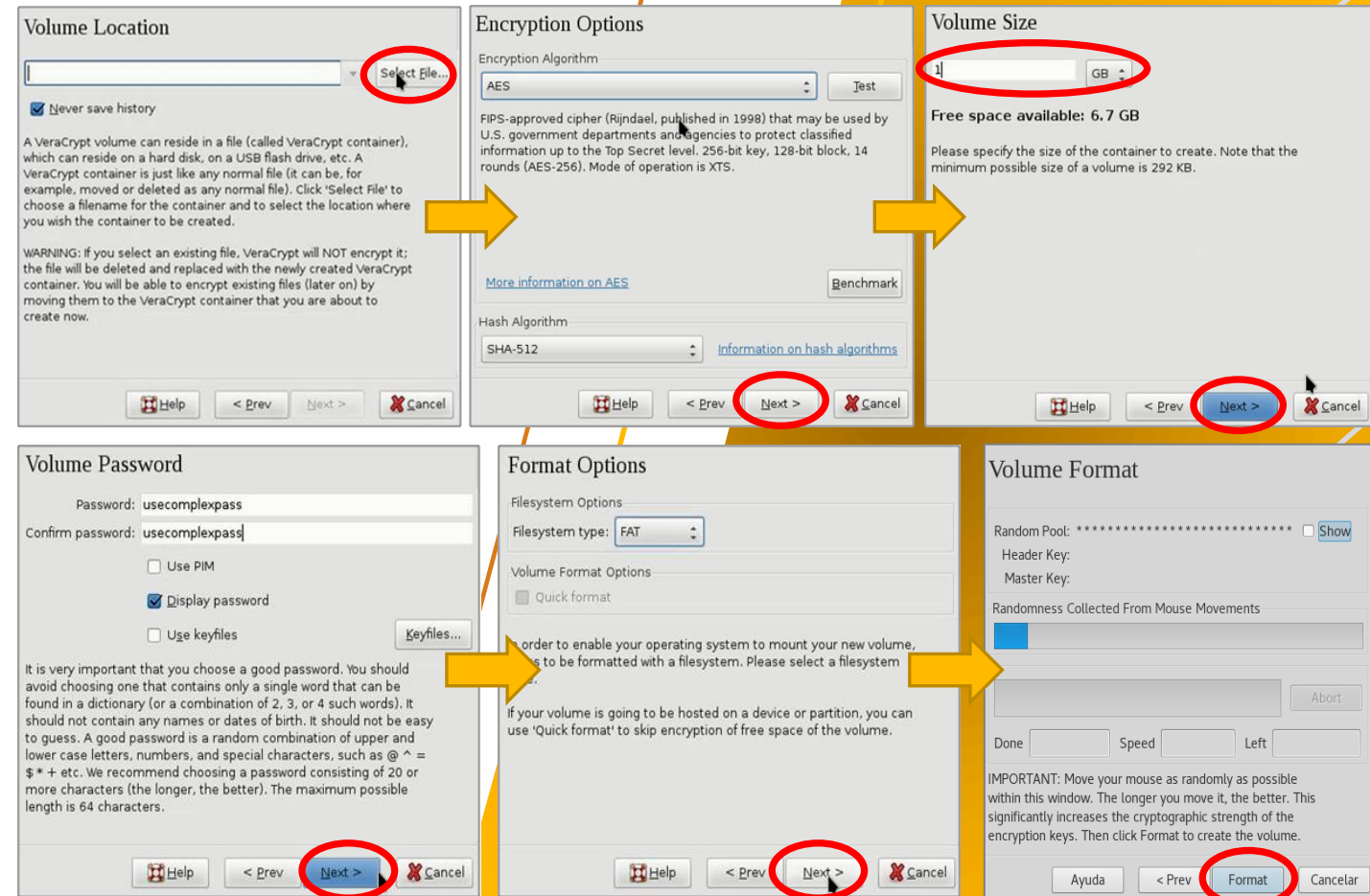
Source: Veracrypt, 2024
 URL: <https://www.veracrypt.fr/code/VeraCrypt/>



Veracrypt – an example of its usefulness

2. Choose the location of the file, the encryption (e.g. AES) and hash (e.g. SHA-512) algorithms, as well as the size of the file
3. Choose the password and the format of the volume contained in the file (e.g. FAT, ExFAT, NTFS...)

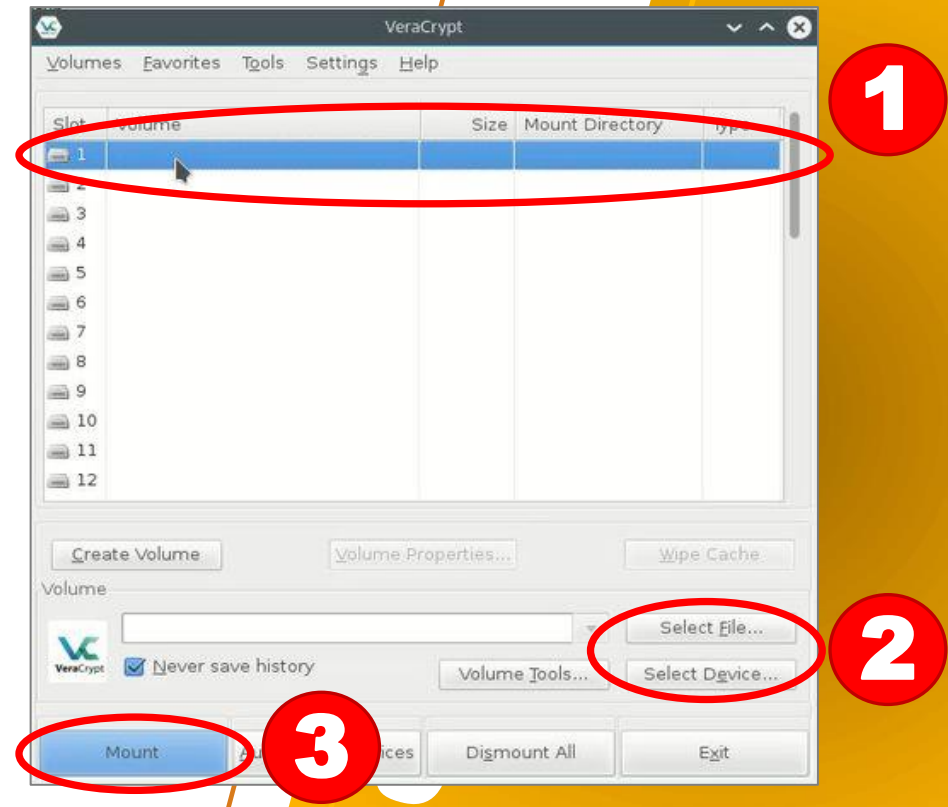
 - At this point, it is required to move the mouse in random directions to create “entropy”, which is necessary for the key generation



Veracrypt - an example of its usefulness

- Example of its use

1. Select a slot
2. Choose the file that contains the encrypted data volume
3. Mount the file, indicating the password



Source: Veracrypt, 2024
URL: <https://www.veracrypt.fr/code/VeraCrypt/>

Empowering skills with OpenSSL

- **OpenSSL** is a file protection alternative
 - Open-Source tool that includes a general-purpose cryptographic library and mechanisms to be applied in command line
 - OpenSSL commands are generic and valid for any platform
- **Commands:**
 - Generic:
 - **\$ openssl help**
 - **\$ openssl list-standard-commands**
 - **\$ openssl list-message-digest-commands**
 - **\$ openssl list-cipher-commands**
 - Symmetric encryption:
 - **\$ openssl enc -cipher-operation mode -in file.txt**
 - **\$ openssl enc -cipher-operation mode -iv IV -K key -in file.txt**
 - Symmetric decryption:
 - **\$ openssl enc -cipher-operation mode -in file.txt -d**
 - **\$ openssl enc -cipher-operation mode -iv IV -K key -in file.txt -d**

```
Standard commands
asn1parse      ca              ciphers         cmp
cms            crl             crl2pkcs7      dgst
dhparam        dsa             dsaparam       ec
ecparam        enc             engine          errstr
fipsinstall    gendsa         genpkey         genrsa
help           info           kdf            list
mac            nseq           ocp             passwd
pkcs12         pkcs7          pkcs8          pkey
pkeyparam      pkeyutl        prime          rand
rehash         req            rsa            rsautl
s_client       s_server       s_time         sess_id
smime          speed          spkac          srp
storeutl       ts             verify         version
x509

Message Digest commands (see the `dgst' command for more details)
blake2b512     blake2s256     md4             md5
mdc2           rmd160         sha1            sha224
sha256         sha3-224       sha3-256       sha3-384
sha3-512       sha384         sha512         sha512-224
sha512-256     shake128        shake256        sm3

Cipher commands (see the `enc' command for more details)
aes-128-cbc    aes-128-ecb    aes-192-cbc    aes-192-ecb
aes-256-cbc    aes-256-ecb    aria-128-cbc   aria-128-cfb
aria-128-cfb1  aria-128-cfb8  aria-128-ctr   aria-128-ecb
aria-128-ofb   aria-192-cbc   aria-192-cfb   aria-192-cfb1
aria-192-cfb8  aria-192-ctr   aria-192-ecb   aria-192-ofb
aria-256-cbc   aria-256-cfb   aria-256-cfb1  aria-256-cfb8
aria-256-ctr   aria-256-ecb   aria-256-ofb   base64
bf             bf-cbc         bf-cfb         bf-ecb
bf-ofb        camellia-128-cbc
camellia-192-ecb
camellia-256-cbc
camellia-256-ecb
cast-cbc      cast5-cbc      cast5-cfb     cast5-ecb
cast5-ofb     des            des-cbc       des-cfb
des-ecb       des-ede        des-ede-cbc   des-ede-cfb
des-ede-ofb   des-ede3       des-ede3-cbc  des-ede3-cfb
des-ede3-ofb  des-ofb        des3          desx
idea          idea-cbc       idea-cfb      idea-ecb
idea-ofb      rc2            rc2-40-cbc    rc2-64-cbc
rc2-cbc       rc2-cfb       rc2-ecb       rc2-ofb
rc4           rc4-40        seed          seed-cbc
seed-cfb      seed-ecb      sm4-cbc       sm4-cbc
sm4-cfb       sm4-ctr       sm4-ecb       sm4-ofb
```

Empowering skills with OpenSSL

- Generate keys:
 - `openssl enc -aes-256-cbc -P -k key`
 - `$ openssl genrsa n-bits > private.key`
 - `$ openssl rsa -in private.key -pubout -out public.pubkey`
 - `$ openssl ecparam -out key.pem -name prime256v1 -genkey`
 - ...
- Asymmetric encryption:
 - `$ openssl rsautl -encrypt -in input.txt -pubin -inkey public.pubkey -out ciphertext.txt`
- Asymmetric decryption:
 - `openssl rsautl -decrypt -in ciphertext.txt -out plaintext.txt -inkey private.key`
- Digital signature:
 - `$ openssl rsautl -sign -in plaintext.txt -out signature.txt -inkey private.key`
- Digital signature verification:
 - `openssl rsautl -verify -in signature.txt -out verification.txt -pubin -inkey public.pubkey`
- Hash
 - `$openssl dgst [-sha | -sha1 | -mdc2 | -ripemd160 | -sha224 | -sha256 | -sha384 | -sha512 | -md2 | -md4 | -md5 | -dss1] [-c] [-d] [-hex] [-binary] [-r] [-hmac arg] [-non-fips-allow] [-out filename] [-sign filename] [-keyform arg] [-passin arg] [-verify filename] [-prverify filename] [-signature filename] [-hmac key] [-non-fips-allow] [-fips-fingerprint] [file...]`

Final remarks

- We have explored the advantages that cryptography can bring to the protection of data in energy control systems, such as:
 - Substations, control networks, corporative networks
- This level of protection can be applied to:
 - **Data in transit:** between industrial equipment, IT devices and users, etc.
 - **Data at rest:** servers, repositories, tablets, HMI, etc.
- For protection, we can apply:
 - Symmetric and asymmetric cryptography, or its combined, for confidentiality
 - Digital certificates for trust
 - Hash functions for integrity
 - MAC functions for authentication and integrity

... amazing, isn't it? 😊



References and sources

1. Netresec, "Captures files from 4SICS Geek Lounge", 2024
URL: <https://www.netresec.com/?page=PCAP4SICS>
2. CS3StHtm, 2014-2020, accessed in 2024.
URL: <https://cs3sthlm.se>
3. CloudShark "telnet-client-server.pcapng", accessed in 2024
URL: <https://www.cloudshark.org/captures/818ceaf07b8?filter=telnet>
4. Buchanan, William J., Human-Readable Encryption Keys. Asecuritysite.com, 2024.
URL: <https://asecuritysite.com/encryption/plain>
5. Wierk, Cryptii, 2024.
URL: <https://cryptii.com>
6. Buchanan, William J., AES. Asecuritysite.com, 2024.
URL: <https://asecuritysite.com/symmetric/aes>
7. Buchanan, William J., DES Cipher. Asecuritysite.com, 2024.
URL: <https://asecuritysite.com/symmetric/des>
8. Buchanan, William J., 3DES Cipher. Asecuritysite.com, 2024.
URL: <https://asecuritysite.com/symmetric/threedes>

References and sources

9. Buchanan, William J., Camellia cipher. Asecuritysite.com. 2024.
URL: <https://asecuritysite.com/symmetric/camellia>
10. Buchanan, William J., Symmetric key modes: GCM, ChaCha20, CBC, CFB8, CFB, OFB, GCM, CTR, and XTS, Asecuritysite.com, 2024.
URL: <https://asecuritysite.com/symmetric/sym>
11. CryptoTool.net, RSA Key Generator, 2024.
URL: <https://cryptotools.net/rsagen>
12. Buchanan, William J, Hashing, Asecuritysite.com, 2024.
URL: <https://asecuritysite.com/encryption/md5>
13. Veracrypt, 2024
URL: <https://www.veracrypt.fr/code/VeraCrypt/>
14. DeepL Translator for Proofreading:
URL: <https://www.deepl.com/translator>
15. Some figures are attributed from Vecteezy,
URL: <https://www.vecteezy.com/> - thanks !



Connect with CyberSecPro: How to register and other practical information

1. Website:
www.cybersecpro-project.eu
2. X (Twitter):
https://twitter.com/CyberSecPro_eu
3. LinkedIn:
<https://www.linkedin.com/company/cybersecpro-euproject/>



**Co-funded by
the European Union**

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or HADEA. Neither the European Union nor the granting authority can be held responsible for them.

Project Agreement no. 101083594

 ACEEU ACCREDITATION COUNCIL FOR ENTREPRENEURIAL & ENGAGED UNIVERSITIES	 AIT AUSTRIAN INSTITUTE OF TECHNOLOGY	 APIROPLUS SOLUTIONS	 SINTEF	 SOCIAL ENGINEERING ACADEMY	 TAL TECH
ACEEU GmbH Germany Visit Website	AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GMBH Austria Visit Website	APIROPLUS SOLUTIONS LTD Cyprus Visit Website	SINTEF AS Norway Visit Website	Social Engineering Academy GmbH Germany Visit Website	Tallin University of Technology Estonia Visit Website
Logo missing	 COFAC COOPERATIVA DE FORMACAO E ENRIAMACAO CULTURAL C.R.L.	 Consiglio Nazionale delle Ricerche	 Technische Universität Braunschweig	 ΠΑΝΤΕΧΝΙΟ ΙΡΑΠΤΗΣ / TECHNICAL UNIVERSITY OF CRETE	 trustilio Enhance your Trustworthiness
C2B CONSULTING Visit Website	COFAC Portugal Visit Website	Consiglio Nazionale delle Ricerche Italy Visit Website	Technical University of Braunschweig Germany Visit Website	Technical University of Crete Greece Visit Website	trustilio B.V. The Netherlands Visit Website
 focal point Cyber Defence Exercises as a Service	 GOETHE UNIVERSITÄT FRANKFURT AM MAIN	 ITML	 LNINOVA	 UNIVERSIDAD DE MÁLAGA	 NOVA UNIVERSIDADE NOVA DE LISBOA
FDAL POINT Belgium Visit Website	Goethe University Frankfurt Germany Visit Website	Information Technology for Market Leadership Greece Visit Website	Uninova Portugal Visit Website	Universidad de Malaga Spain Visit Website	Universidade Nova De Lisboa Portugal Visit Website
 Institut Mines-Télécom	 LAUREA	 GRUPPO Maggioli	 University of Cyprus	 FACULTY OF SCIENCES NOVI SAD 1969 SERBIA	 UNIVERSITY OF PIRAEUS RESEARCH CENTER
Institut Mines-Télecom France Visit Website	Laurea University of Applied Sciences Finland Visit Website	Maggioli S.p.A. Italy Visit Website	University of Cyprus Cyprus Visit Website	University of Novi Sad Faculty of Sciences Serbia Visit Website	University of Piraeus Research Center Greece Visit Website
 PDMFC	 Security Labs Consulting Ltd	 SGI	 Zelus		
PDMFC Portugal Visit Website	Security Labs Consulting Ltd Ireland (Republic) Visit Website	Serious Games Interactive Denmark Visit Website	ZELUS P.C. Greece Visit Website		

Thank you

If you have any questions, please do not hesitate to contact:

- Davide Ferraris
Substitute Professor
University of Malaga
ferraris@uma.es