Cryptography

The NTUA_H4CK Workshop

- Γιάννης Κουμπιάς
- Χρόνης Σαπουντζάκης
- Σήλια Κοντοθανάση
- Μάρω Τερζή

Challenge Coded by:

- Στυλιανός Αναστασίου

- 180365595049195148200891720356467438656786937306378809265211771932588396622516255630083059247834973205931722041854674277468 3

- 65537

- 613771544420165681333550699569249438364853505684828774175718889969910774297514578985929583747274623357178499089846261339341

Γιατί κρυπτογραφία;

- Η κρυπτογραφία αποτελεί θεμελιώδη κλάδο της κυβερνοασφάλειας στη σύγχρονη εποχή, προσφέροντας μυστικότητα στη μετάδοση και διατήρηση δεδομένων.

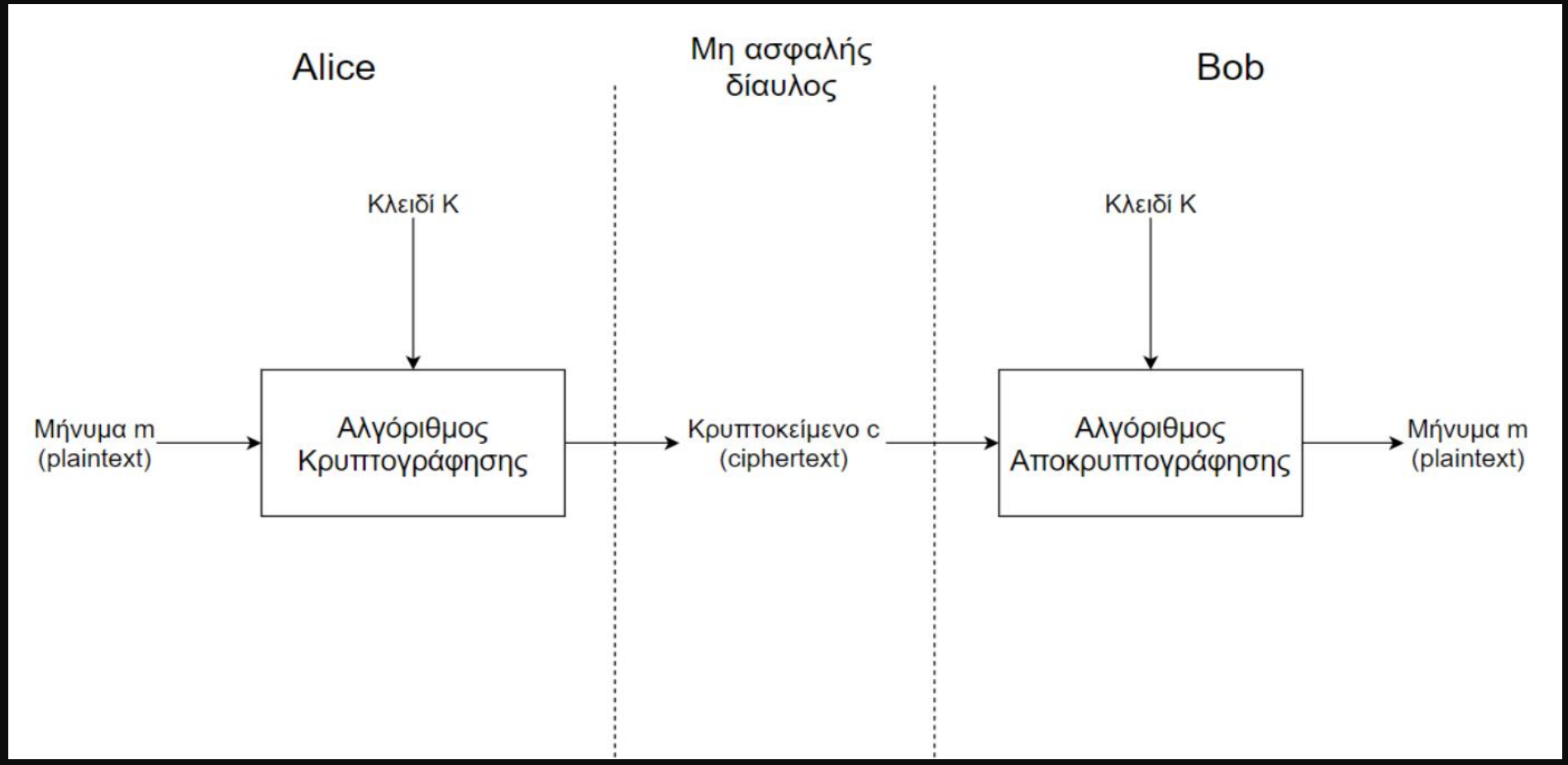
- Οι βασικοί στόχοι της κρυπτογραφίας είναι οι:
  - **Εμπιστευτικότητα (Confidentiality)**
  - **Ακεραιότητα (Integrity)**
  - **Πιστοποίηση (Authentication)**

- Πως όμως πετυχαίνει η κρυπτογραφία τα παραπάνω;

Αλγόριθμοι Κρυπτογράφησης

- Ένα άτομο Α (Alice) επιθυμεί να επικοινωνήσει με ένα άτομο Β (Bob) μέσω ενός μη ασφαλή διαύλου.
- Η Alice κρυπτογραφεί το μήνυμα της m μέσω ενός αλγορίθμου που χρησιμοποιεί ένα κλειδί k_1.
- Το κρυπτοκείμενο c μεταδίδεται στον Bob μέσω του διαύλου επικοινωνίας.
- Ο Bob αποκρυπτογραφεί το μεταδιδόμενο μήνυμα c χρησιμοποιώντας έναν αλγόριθμο με κάποιο κλειδί k_2 και καταλήγει με το αρχικό μήνυμα m.

Αλγόριθμοι Κρυπτογράφησης



Κλασσικοί Αλγόριθμοι Κρυπτογράφησης

Caesar's Cipher

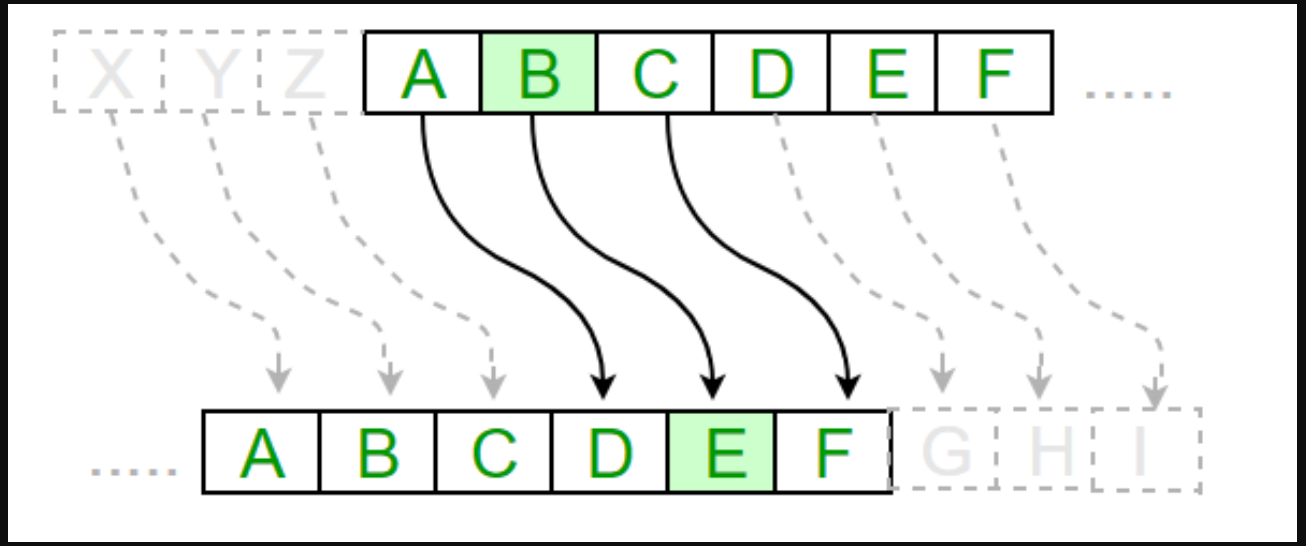
Ο αλγόριθμος του Καίσαρα είναι ένας πολύ απλός αλγόριθμος αντικατάστασης, ο οποίος κάνει shift το αλφάβητο κατά 3 θέσεις:

- A→D
- CRYPTO→FUBSWR

Αντιστοιχίζοντας τα γράμματα A-Z με αριθμούς 0-25 ο αλγόριθμος μπορεί να υλοποιηθεί προσθέτοντας το 3 mod 26.

Μπορούμε να επεκτείνουμε την κρυπτογράφηση κάνοντας shift n φορές προσθέτοντας n mod 26. Τότε έχουμε ένα rotn cipher με κλειδί k=n.

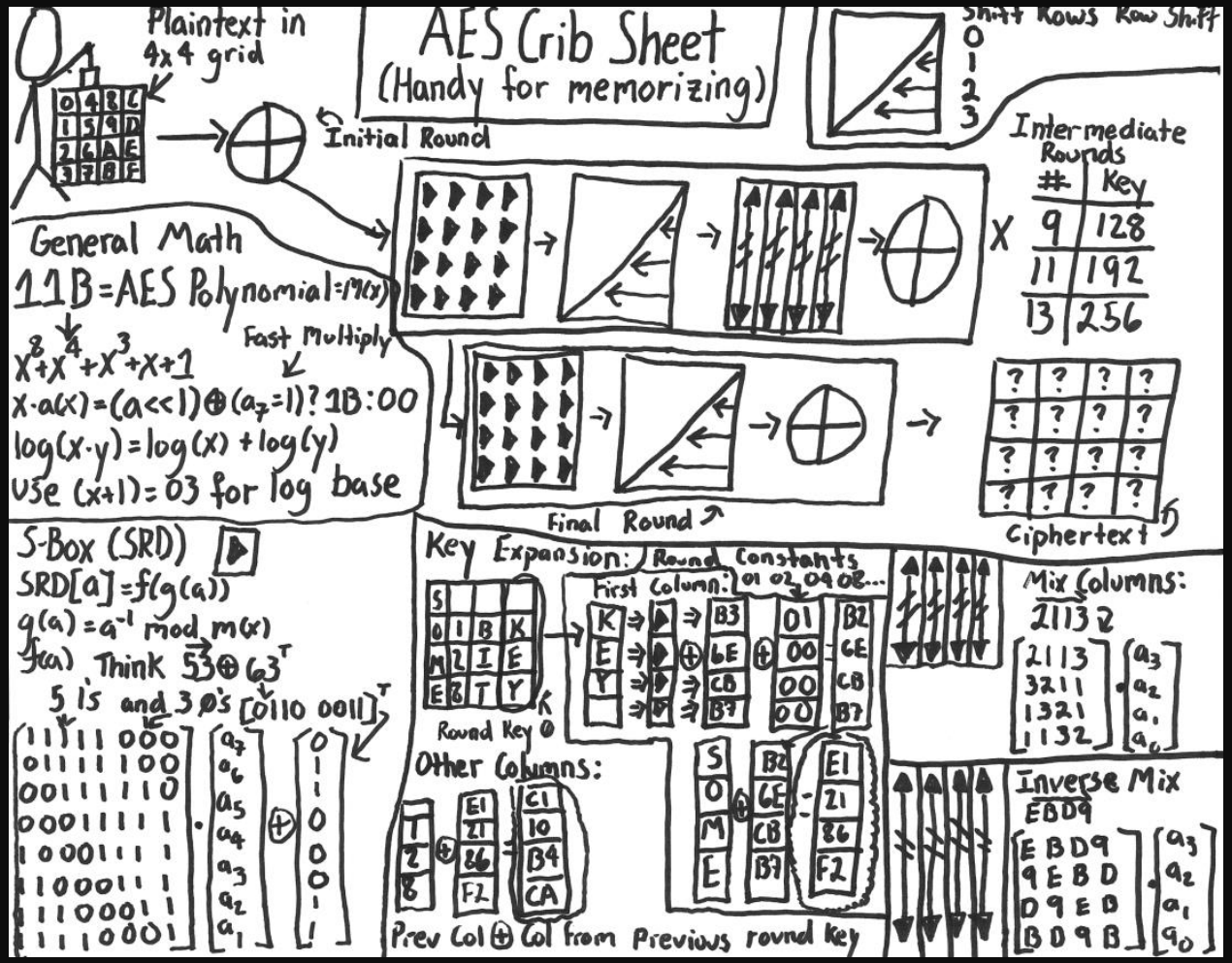
- https://rot13.com



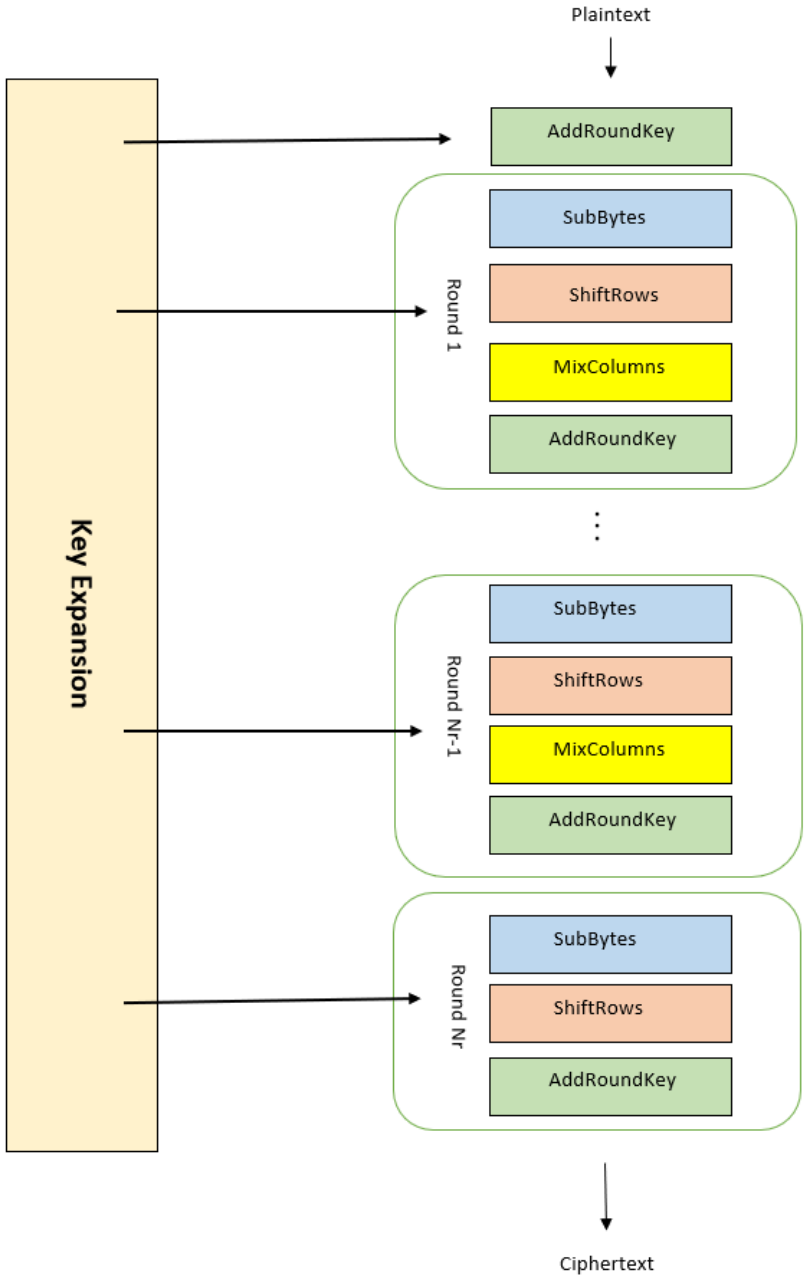
Σύγχρονοι Αλγόριθμοι Κρυπτογράφησης

Advanced Encryption Standard (AES)

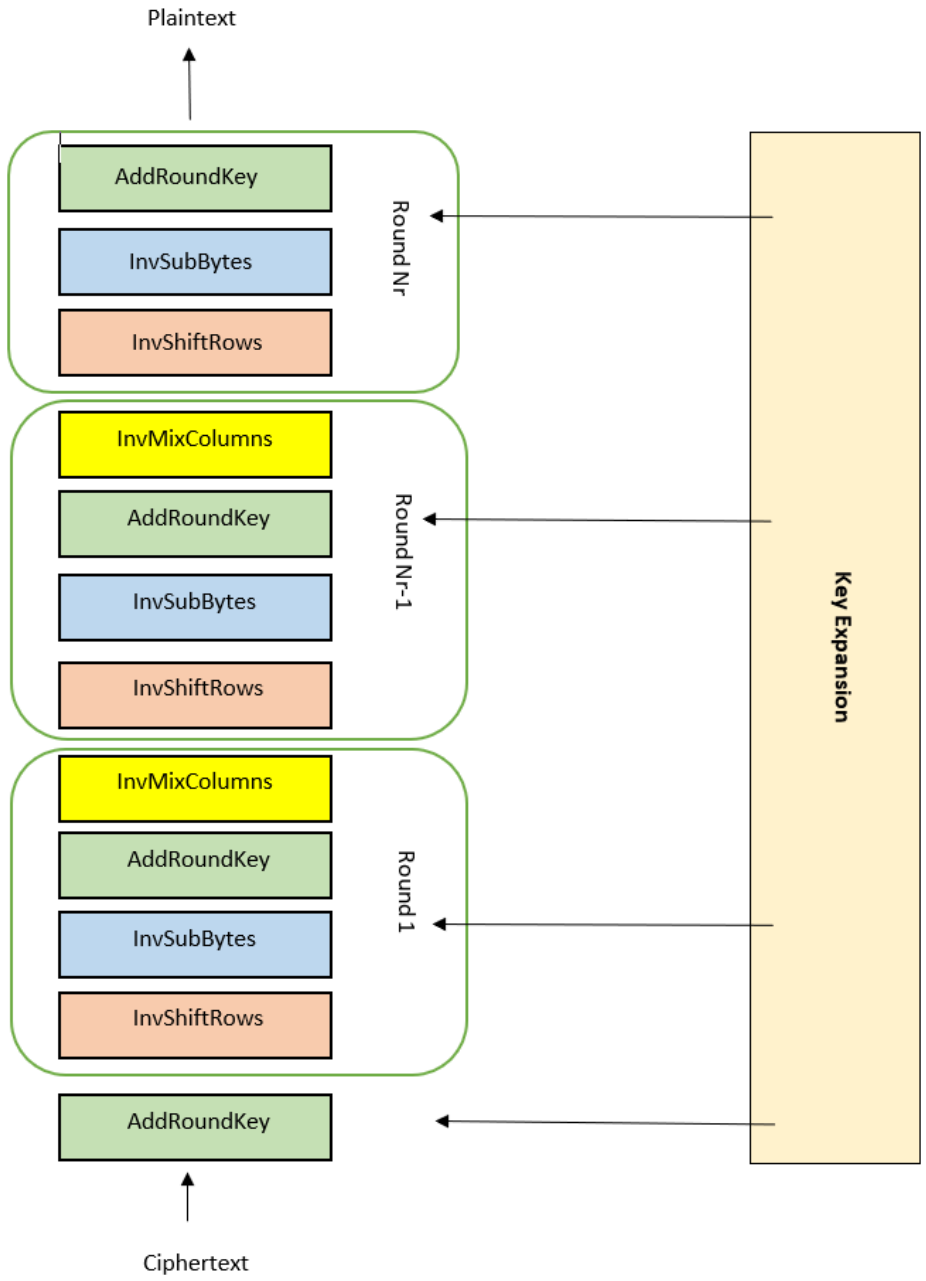
- Αλγόριθμος κρυπτογράφησης συμμετρικού κλειδιού.
- Block Cipher: Τα δεδομένα χωρίζονται σε blocks.
- Different modes: ECB, CBC…
- Μπορεί να μετατραπεί σε stream cipher: CTR…



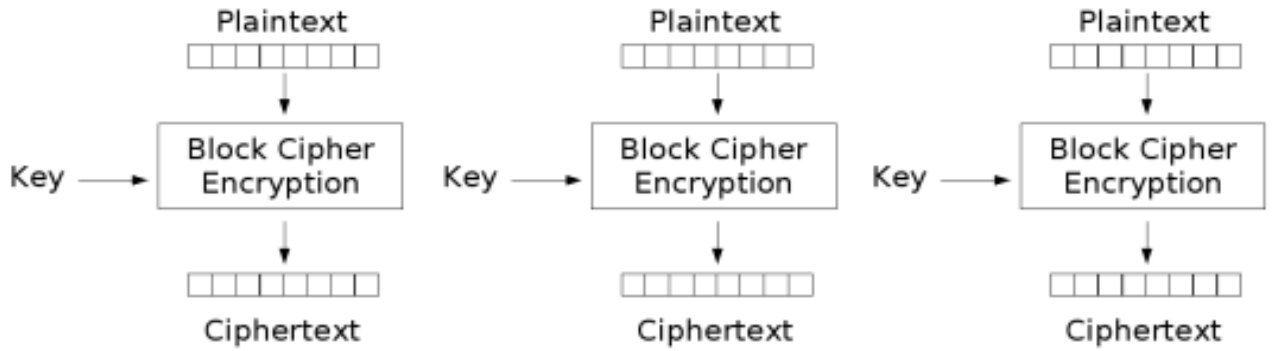
AES – Black Box: Encryption



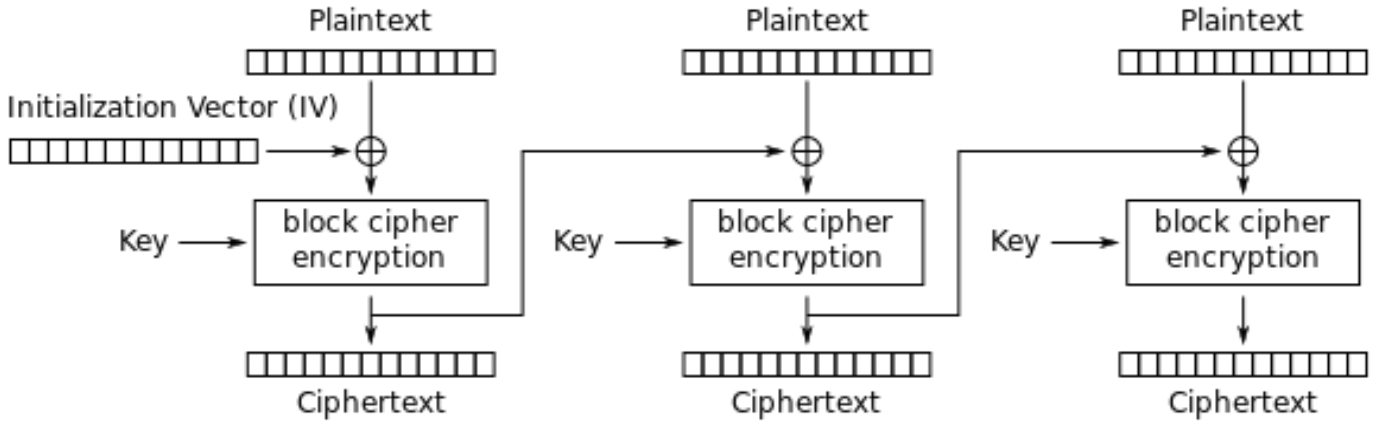
AES – Black Box: Decryption



AES Modes



Electronic Codebook (ECB) mode encryption

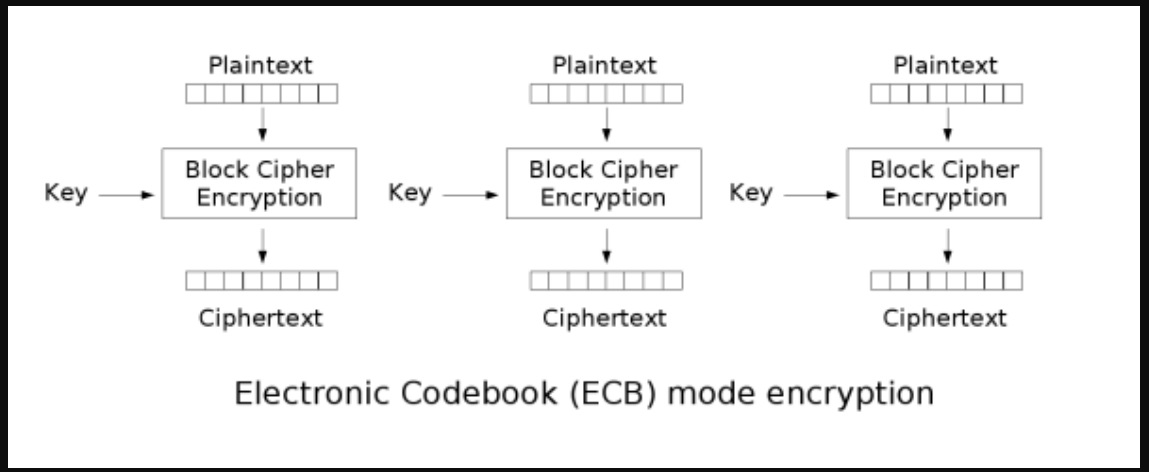


Cipher Block Chaining (CBC) mode encryption

Γιατί όχι ECB;

- Το ECB είναι το πιο απλό mode του AES.
- Το plaintext χωρίζεται σε blocks των 16 bytes.
- Κάθε block γίνεται encrypt με το black box.

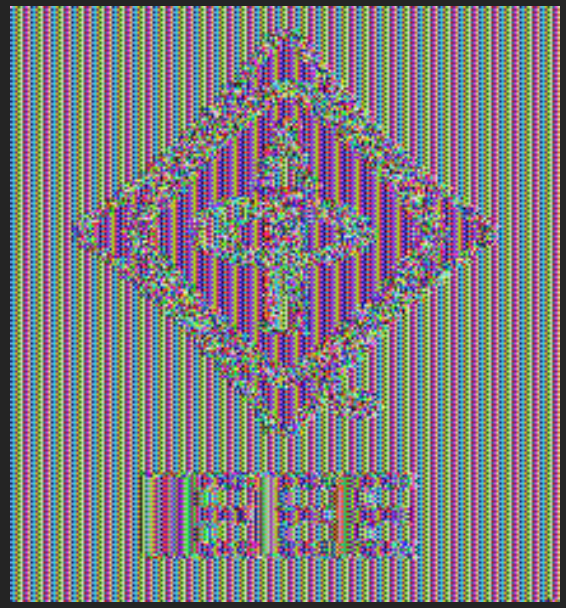
- Γιατί αυτό δεν είναι ασφαλές;
  - LACK OF DIFFUSION!



Before ECB

Γιατί όχι ECB;

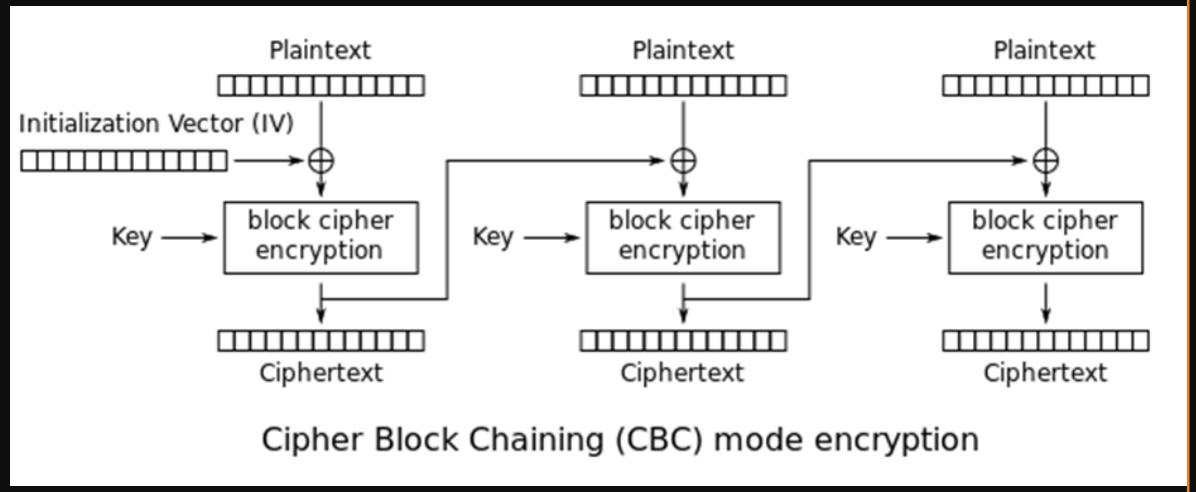
After ECB



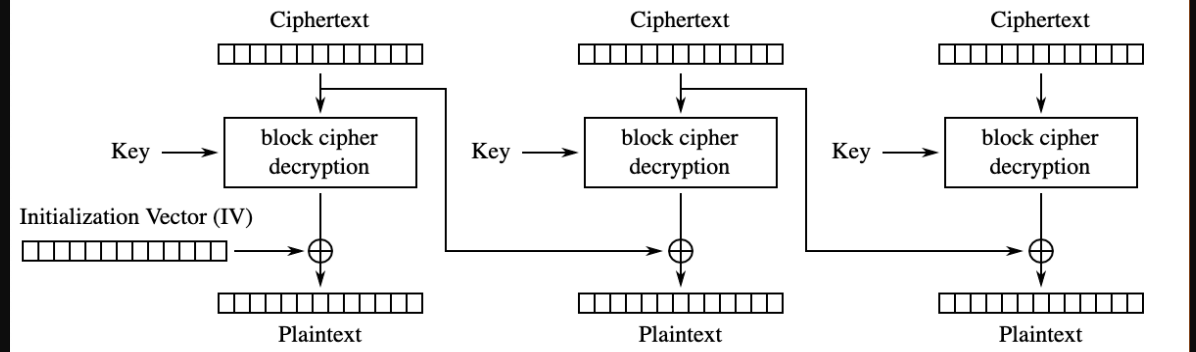
Τι γίνεται με το CBC;

- Στο CBC το plaintext χωρίζεται πάλι σε blocks των 16 bytes.
- PADDING: Αν το τελευταίο block δεν έχει ακριβώς 16 bytes τότε προσθέτω έξτρα bytes μέχρι να συμπληρωθούν!
- Κάθε block του plaintext γίνεται XOR με το προηγούμενο block του ciphertext.
- Το πρώτο block γίνεται XOR με ένα τυχαίο IV (Initialization Vector).
- Ακολουθεί encryption με το black box.

- Γιατί είναι πιο ασφαλές;
  - Κάθε μπλοκ επηρεάζεται από το προηγούμενο.
  - Αλλαγή σε κάποιο bit ενός μπλοκ οδηγεί σε αλλαγές στα bits των επόμενων block!
  - Μοναδικότητα χάρη στο IV.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Before CBC

Τι γίνεται με το CBC;

After CBC

Σύγχρονοι Αλγόριθμοι Κρυπτογράφησης

Rivest, Shamir and Adleman – the creators of RSA

RSA

- Αλγόριθμος κρυπτογράφησης ασύμμετρου κλειδιού
- Στηρίζεται στη δυσκολία του factoring problem
- Ζεύγος Δημοσίου Κλειδιού:
  - Public modulus n
  - Public exponent e
- Ιδιωτικό Κλειδί:
  - Private key d

RSA Προετοιμασία του Bob

- Ο Bob επιλέγει δύο πρώτους : p, q
- Υπολογίζει το public modulus:

$$n = p * q$$

- Υπολογίζει την συν. Euler :

$$\varphi(\, n\,) = (p-1) * (q-1)$$

- Επιλέγει public exponent e (συνήθως 3 ή 65537)
- Υπολογίζει το Private Key d από :

$$ed = 1\ (mod\ \varphi(\, n\,))$$

- Τέλος στέλνει το Public Key (n, e) στην Alice.

RSA Κρυπτογράφηση της Alice

- Η Alice θέλει να στείλει ένα μήνυμα m στον Bob
- Έχει τα (n, e)
- Παράγει το κρυπτογραφημένο μήνυμα c ως:

$$c = m^e \, (mod \, n)$$

- Στέλνει το c στον Bob.

RSA Αποκρυπτογράφηση του Bob

- Ισχύει:

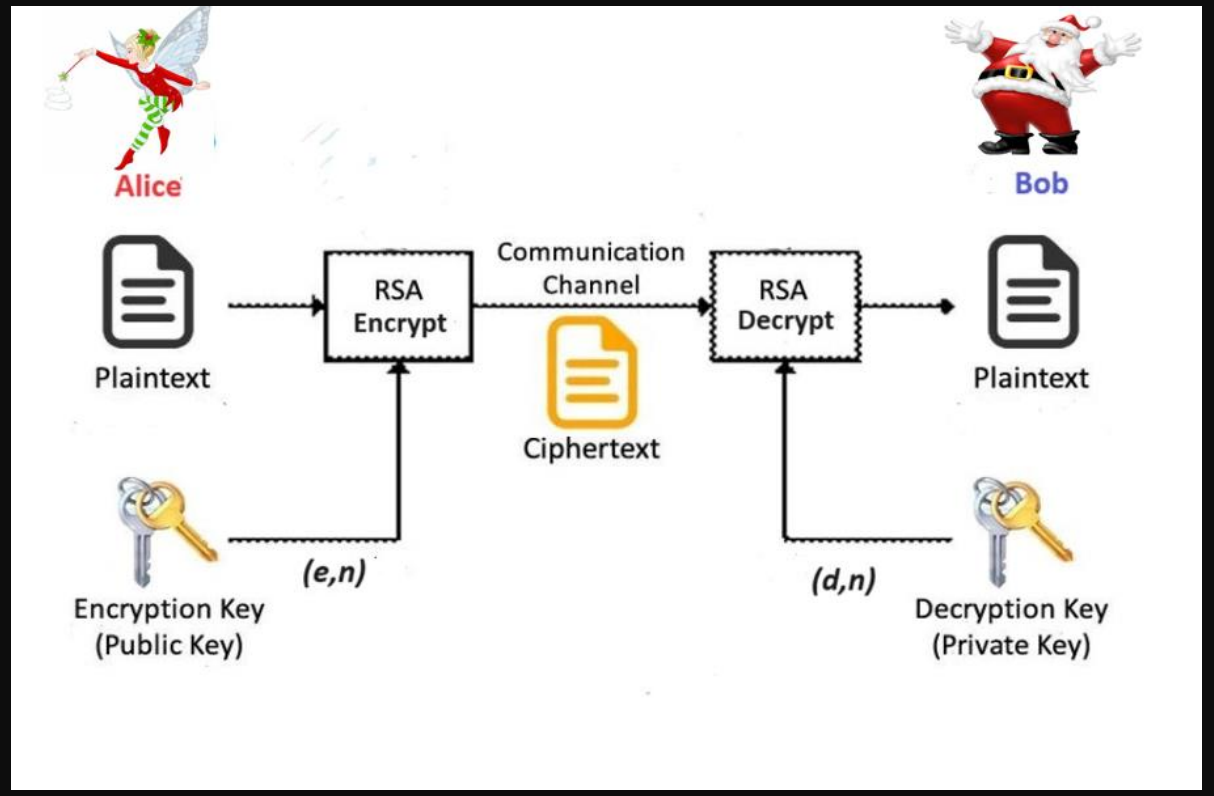
$$ed = 1\ (mod\ \varphi(n))$$

- Άρα ο Bob ανακτά το αρχικό μήνυμα m από:

$$m = c^d\ (mod\ n)$$

- Παρατηρούμε λοιπόν πως:

$$c^d = (m^e)^d = m^{ed} = m\ (mod\ n)$$



RSA Παρατηρήσεις

- Παρατηρούμε πως πρέπει να ξέρουμε το d για να βρούμε το αρχικό μήνυμα.
- Δηλαδή να υπολογίσουμε τον αντίστροφο του $e\ (mod\ \varphi(n))$.

- Ορίζουμε αντίστροφο b ενός αριθμού $a\ mod\ n$ ως:

  $ab = 1\ (mod\ n) \Rightarrow b = a^{-1}(mod\ n)$
- Προσοχή! Ο $a^{-1}(mod\ n)$ δεν είναι κλάσμα αλλά ακέραιος: πχ $3 * 5 = 1\ (mod\ 14)$

- Μόνο ο Bob μπορεί να αποκρυπτογραφήσει το μήνυμα της Alice γιατί μόνο εκείνος έχει το Private Key d.
- Μπορεί η Alice να αποκρυπτογραφήσει μήνυμα που της στέλνει ο Bob και αν ναι πώς;

RSA παράδειγμα σε Python

- Χρήσιμη βιβλιοθήκη: https://pypi.org/project/pycryptodome/3.17/

```
from Crypto.Util.number import getPrime,bytes_to_long,long_to_bytes
```

- Με την getPrime() παίρνουμε 2 πρώτους αριθμούς

```
p=getPrime(128)
q=getPrime(128)
```

- Τα functions bytes_to_long() και long_to_bytes() χρησιμοποιούνται για την μετατροπή bytes σε ακεραίους και ακεραίους σε bytes

RSA Παράδειγμα σε Python

- Υπολογίζουμε τα $n, \varphi(n), e$:

```
n=p*q
phi=(p-1)*(q-1)
e=0x10001 #65537
```

- Το μήνυμα m της Alice είναι “MERRY CHRISTMAS” και το μετατρέπουμε σε ακέραιο ώστε να βρούμε το c

```
m=bytes_to_long(b'MERRY CHRISTMAS')
c=pow(m,e,n)
```

- Ο Bob αποκρυπτογραφεί το μήνυμα c της Alice

```
d=pow(e,-1,phi)
decrypted=pow(c,d,n)
```

RSA παράδειγμα σε Python

```
p: 23667971318117181114578076582983266870 7
q: 22361810410881942418608738396948707193 7
n: 52925868742592799011568995252988600309792561544334085131689559768415997775459
phi(n): 52925868742592799011568995252988600309332263727044093896357691618616678034816
e: 65537
Plaintext as integer: 401212866564677422498393988526850387
Plaintext as bytes: b'MERRY CHRISTMAS'
Ciphertext as integer: 45148978488253327939936641253294313170997500413148095964175386065146086861073
Ciphertext as bytes: b'c\xd1j\x8f\x0f\x0f\xb1R\x9f\xc1gJ\x81\xeb\x129\xe9a\xb2*\xd0\xdd\x06\xfc\xbb\x1b\x16DS\x86\xa9\x11'
Decrypted message: b'MERRY CHRISTMAS'
```

RSA vulnerabilities

- Ο υπολογισμός του ιδιωτικού d προϋποθέτει την γνώση των p, q για τον υπολογισμό του $\varphi(n)$.
- ΔΥΣΚΟΛΟ μόνο με την γνώση του n (https://en.wikipedia.org/wiki/Integer_factorization)
- Άλλοι τρόποι;

- Θα παρουσιάσουμε μερικές περιπτώσεις όπου το RSA είναι ευάλωτο.

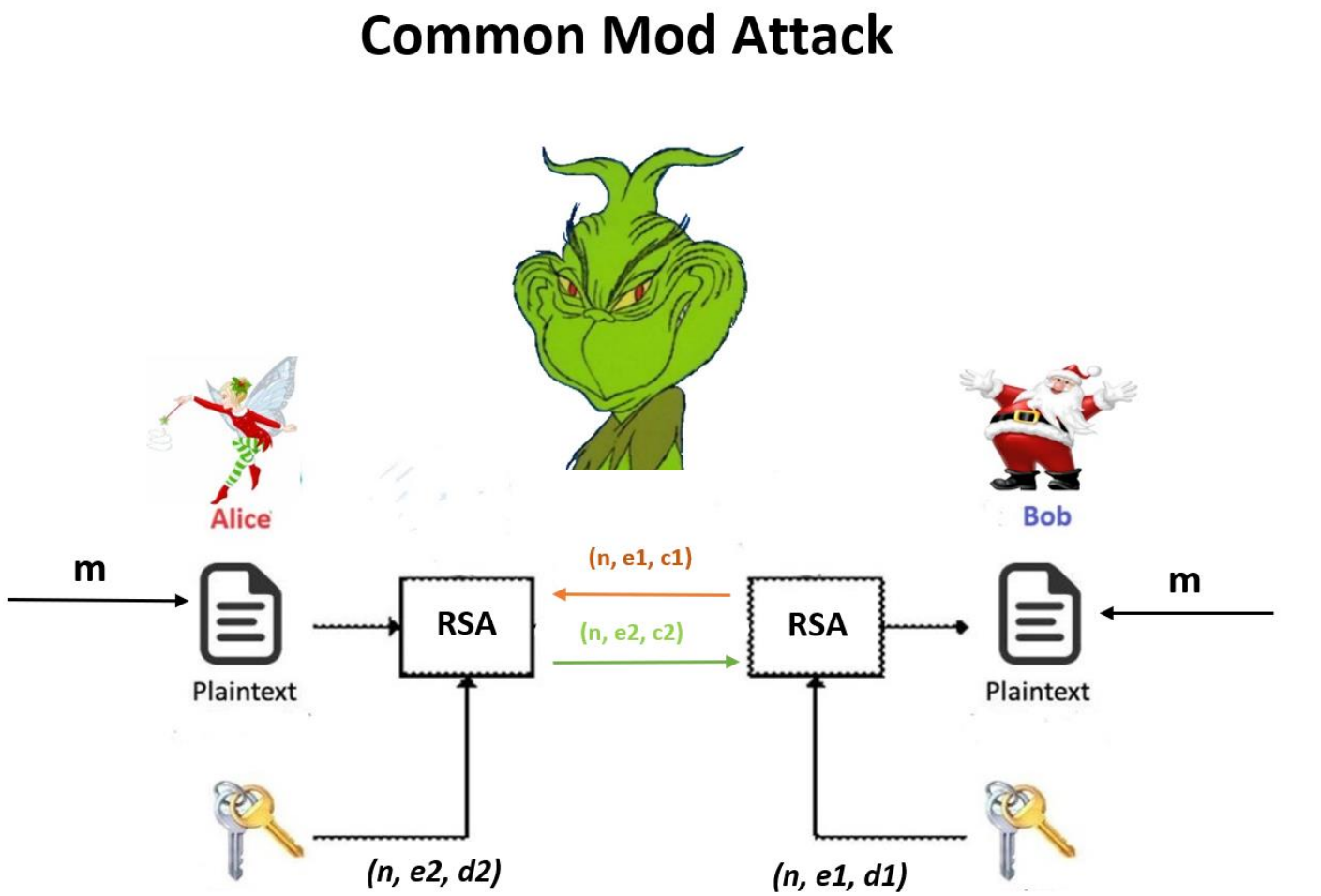
Common Modulus Attack

- Έστω ότι ο Bob και η Alice θέλουν να συνεννοηθούν για την διανομή των δώρων των Χριστουγέννων.
- Ο Bob στέλνει το public key (n, e_1)
- Η Alice στέλνει το public key (n, e_2)

- Για να επικοινωνήσουν στέλνουν και οι δύο το μήνυμα:

$$m = \text{“the presents shall be delivered by dawn”}$$

Common Modulus Attack



Common Modulus Attack

- Η Alice και ο Bob έχουν:
  - το ίδιο n και
  - διαφορετικό e!
- 2η σημαντική προϋπόθεση:

$$\gcd(e_1, e_2) = 1$$

- Αν ισχύει κάτι τέτοιο ο Grinch έχει:

$$c_1 = m^{e_1} \ (mod\ n)$$

$$c_2 = m^{e_2} \ (mod\ n)$$

οπού c_1 και c_2 το κρυπτογραφημένο μήνυμα με δύο διαφορετικά e.

Common Modulus Attack

Ιδέα επίθεσης:

- Έστω ακέραιοι x, y.

  Υπολογίζω το : $c_1^x * c_2^y \ (mod\ n)$

- Τότε θα ισχύει:

$$c_1^x * c_2^y = m^{xe_1} * m^{ye_2} = m^{xe_1 + ye_2} \ (mod\ n)$$

ΕΡΩΤΗΣΗ: Πώς βρίσκω κατάλληλα x, y ώστε $xe_1 + ye_2 = 1$;

- **Επεκτεταμένος Αλγόριθμος Ευκλείδη** (Extended Euclidean Algorithm)
- Ιδέα αλγορίθμου: Αξιοποίηση των πηλίκων στον Κλασσικό Αλγόριθμο Ευκλείδη
- https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

Common Modulus Attack - Python

- Στήνουμε ένα RSA με $e_1 = 11, e_2 = 13$
- Κρυπτογραφούμε το μήνυμα m δύο φορές.
- Υπολογίζουμε τα x, y με το function egcd().

```
e1=11
e2=13
p=getPrime(256)
q=getPrime(256)
n=p*q
```

```
m=bytes_to_long(b'the presents shall be delivered by dawn')
c1=pow(m,e1,n)
c2=pow(m,e2,n)
```

```
gcd,y,x=egcd(e2,e1)
print(f"Bezout Coefficients: {x} , {y}")
```

- Βρίσκουμε το μήνυμα.

```
decrypted=(pow(c1,x,n)*pow(c2,y,n))%n
```

Common Modulus Attack - Python

```
Cipher1: 26603319610243302831152687468700131499808045197233936516247354928197992141869589016630402462646059474404698244734476536346569792545478603576447991215600 53
Cipher2: 57400437196515005680173711867552882391855091203019173399143877368767545536593092338903860262597413119748629491046645530153107141965152948078762620013294 3
Bezout Coefficients: 6 , -5
Grinch knows that the presents shall be delivered by dawn
```

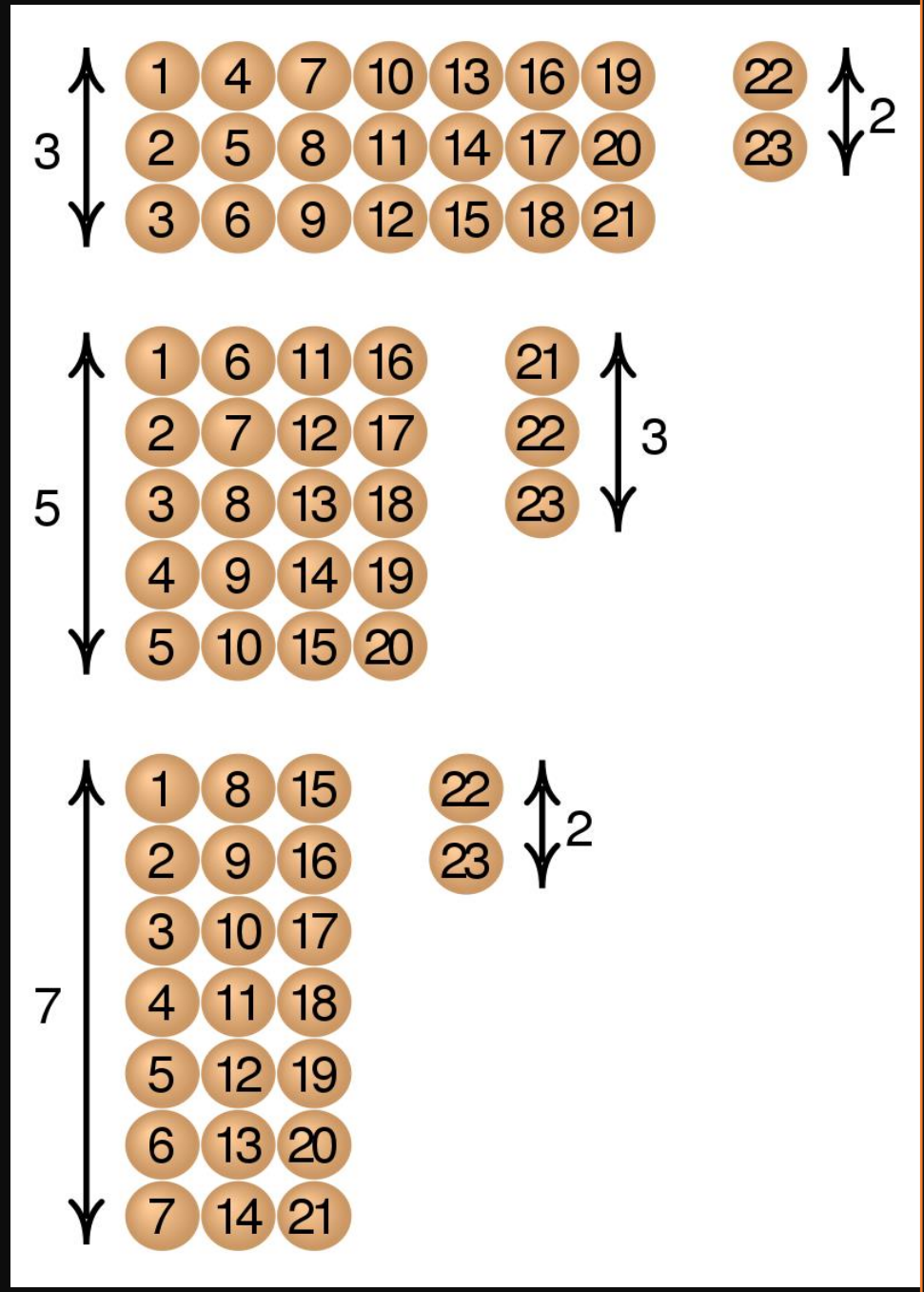
Chinese Remainder Theorem(**CRT**)

Παράδειγμα :

Έστω ότι γνωρίζω ότι :

$$x = 0 \pmod 3$$
$$x = 0 \pmod 5$$

- Πόσο είναι το x;



"There are certain things whose number is unknown. If we count them by threes, we have two left over; by fives, we have three left over; and by sevens, two are left over. How many things are there?" - Sunzi

Chinese Remainder Theorem(**CRT**)

Παράδειγμα :

Έστω ότι γνωρίζω ότι :

$$x = 0 \pmod 3$$
$$x = 0 \pmod 5$$

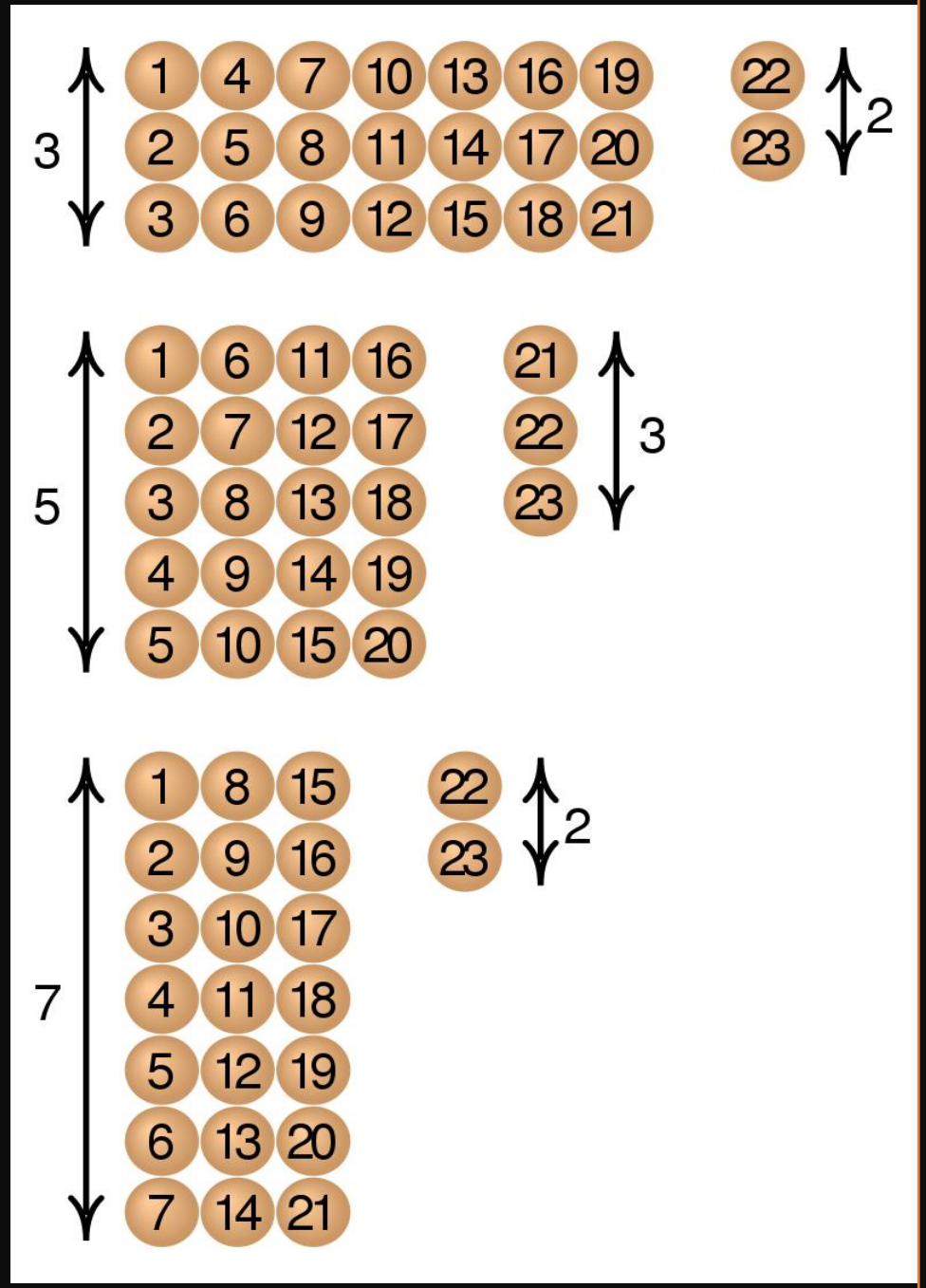
- Πόσο είναι το x;

$x = 0$ ή $x = 15$ ή $x = 30$

Γενικά:

$$\boldsymbol{x = 15k + 0}$$

Παρατήρηση: $15 = 3 * 5$, ΚΑΘΟΛΟΥ τυχαίο



"There are certain things whose number is unknown. If we count them by threes, we have two left over; by fives, we have three left over; and by sevens, two are left over. How many things are there?" - Sunzi

Chinese Remainder Theorem(**CRT**)

Λίγο πιο δύσκολο παράδειγμα:

Έστω ότι γνωρίζω ότι :

$$x = 1 \pmod 3$$
$$x = 0 \pmod 5$$

- Μπορείτε να βρείτε το x;

Chinese Remainder Theorem(**CRT**)

Λίγο πιο δύσκολο παράδειγμα:

Έστω ότι γνωρίζω ότι :

$$x = 1 \pmod 3$$
$$x = 0 \pmod 5$$

- Μπορείτε να βρείτε το x;

$$x = 10 \text{ ή } x = 25$$

Γενικά:

$$x = 15k + 10$$

Chinese Remainder Theorem(**CRT**)

Γενικά, με k εξισώσεις:

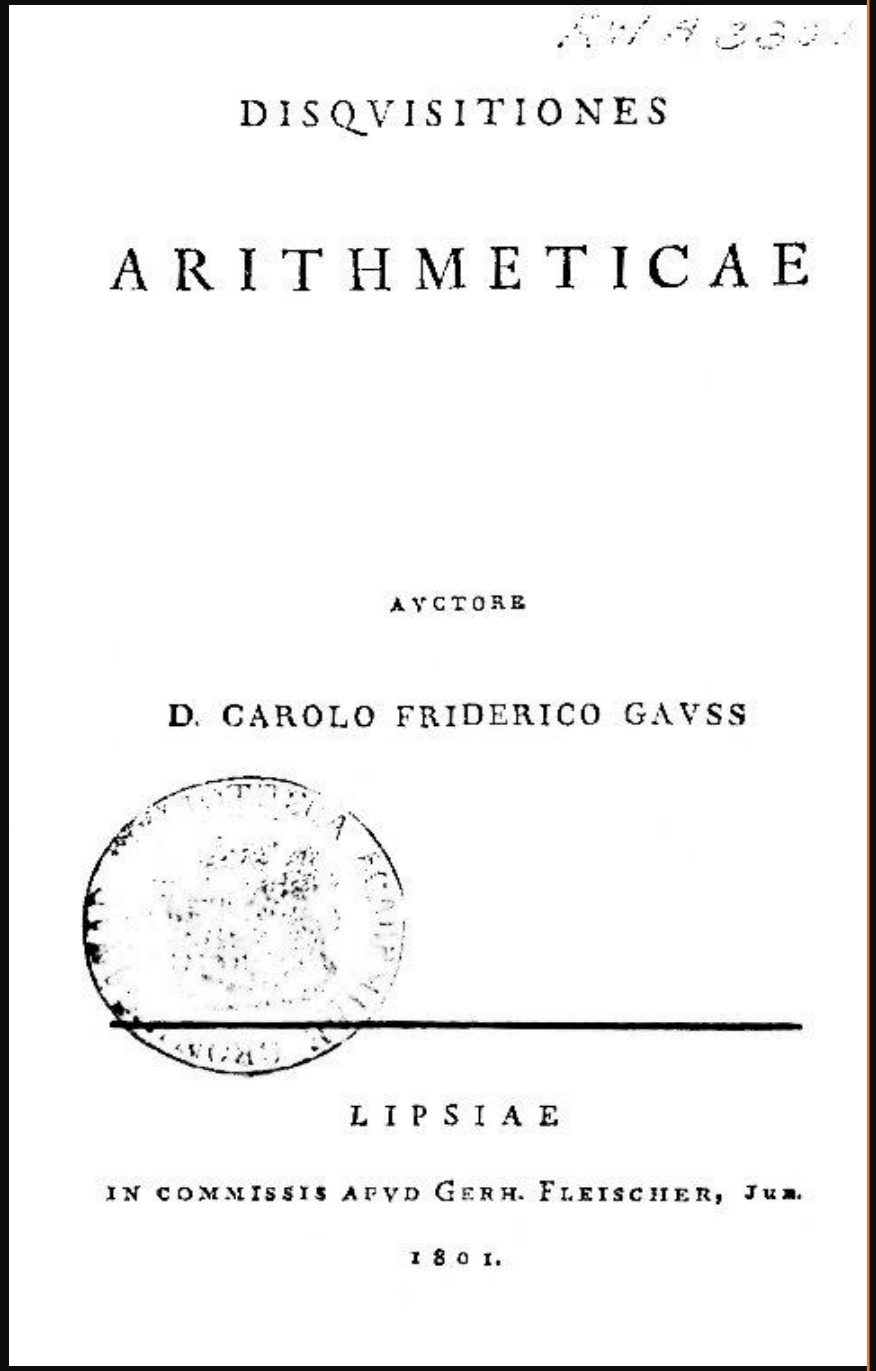
$$x = a_1 \pmod{p_1}$$
$$x = a_2 \pmod{p_2}$$
$$\vdots$$
$$x = a_k \pmod{p_k}$$

όπου : $\boldsymbol{a_i}$ **το υπόλοιπο** τα διαίρεσης του αριθμού x με το αντίστοιχο $\mathbf{p_i}$.

Μπορούμε να βρούμε το :

$$x \left(mod\ \prod_{i=1}^{k} p_i\right) \text{ !!!}$$

Με την προϋπόθεση ότι οι αριθμοί $\mathbf{p_i}$ είναι **πρώτοι μεταξύ τους**.


DISQVISITIONES

ARITHMETICAE

AVCTORE

D. CAROLO FRIDERICO GAVSS

LIPSIAE

IN COMMISSIS APVD GERH. FLEISCHER, JUN.

1801.

RSA Hastad's Broadcast Attack

Έστω ότι έχουμε **τρία public keys** όλα με ίδιο $e = 3$ αλλά διαφορετικό n

$$(\mathbf{3}, \boldsymbol{n_1}), \qquad (\mathbf{3}, \boldsymbol{n_2}), \qquad (\mathbf{3}, \boldsymbol{n_3})$$

Αν η Alice στείλει το ίδιο **μήνυμα** $\boldsymbol{m}$ και με τα τρία αυτά public keys.

- $\boldsymbol{c_1} = m^3 \ (mod\ n_1) \Rightarrow \boldsymbol{c_1} = m^3 - k_1 n_1$
- $\boldsymbol{c_2} = m^3 \ (mod\ n_2) \Rightarrow \boldsymbol{c_2} = m^3 - k_2 n_2$
- $\boldsymbol{c_3} = m^3 \ (mod\ n_3) \Rightarrow \boldsymbol{c_3} = m^3 - k_3 n_3$

όπου: $\boldsymbol{c_i}$ **το i κρυπτογραφημένο μήνυμα**

Αν είχαμε το m^3 (και όχι reduced mod κάποιο n) θα μπορούσαμε απλά να υπολογίσουμε την **κυβική ρίζα** του και θα παίρναμε το m.

Μααα … αυτό είναι CRT

Το x είναι το m^3 και έχουμε τις τρεις εξισώσεις:

- $x = \boldsymbol{c_1} \pmod{n_1}$
- $x = \boldsymbol{c_2} \pmod{n_2}$
- $x = \boldsymbol{c_3} \pmod{n_3}$

➢ Άρα με CRT μπορούμε να βρούμε το $x = \boldsymbol{m}^3 \ (mod\ n_1 n_2 n_3)$ δηλαδή το **σωστό** $\boldsymbol{m}^3$

Λίγες λεπτομέρειες

Γιατί χρειαζόμαστε 3 public keys(όσα το e)?

Αν m **αρκετά** μικρό τότε $m^3 < n_i$ οπότε μπορούμε **κατευθείαν** να βρούμε την κυβική ρίζα

Αλλά για μεγάλο m δεν μπορούμε. Όμως ξέρουμε ότι:

- $m < n_1$
- $m < n_2$
- $m < n_3$

Άρα $m^3 < n_1 n_2 n_3$, (πολλαπλασιάζοντας κατά μέλη)

Λίγες ακόμα λεπτομέρειες

Είπαμε για να κάνουμε CRT πρέπει τα n_1, n_2, n_3 να είναι **μεταξύ τους πρώτα**.

- Αν **δεν** είναι;

Left as an exercise to the reader.

Το παράδειγμα σε python

- Φτιάχνουμε 3 κλειδιά

```python
from Crypto.Util.number import getPrime, long_to_bytes, bytes_to_long

def KeyGen():
    p, q = getPrime(512), getPrime(512)
    n = p*q
    return (n, 3)

n1, e = KeyGen()
n2, e = KeyGen()
n3, e = KeyGen()
```

- Κάνουμε encrypt το ίδιο μήνυμα και με τα 3

```python
msg = b"Wow really long message hopefully long enough so you can't immediatelly take the cubic root but smaller than n1, n2, n3"
m = bytes_to_long(msg)

c1 = pow(m, e, n1)
c2 = pow(m, e, n2)
c3 = pow(m, e, n3)
```

```python
from sympy.ntheory.modular import crt # import crt from sympy
recovered_m_raised_to_3, n1n2n3 = crt([n1, n2, n3], [c1, c2, c3]) # returns (m^3, n1*n2*n3)
```

Το παράδειγμα σε python

- Με CRT βρίσκουμε το m^3
- Βρίσκουμε κυβική ρίζα του m^3 (δηλαδή το m!)

```python
from gmpy2 import iroot
recovered_m, _ = iroot(recovered_m_raised_to_3, 3) # returns (cubic_root, True) if input is a perfect cube (approximate_cubic_root, False) otherwise
print(long_to_bytes(recovered_m))
```

Άλλες επιθέσεις...

- Low private exponent (Wiener Attack) https://pypi.org/project/owiener/1.0.6/
- Low public exponent
- Coppersmith related attacks https://en.wikipedia.org/wiki/Coppersmith%27s_attack
- Implementation Attacks
- Βλ. Χρήσιμο pdf στο υλικό

When all else fails…

- Αν θέλουμε να παραγοντοποιήσουμε το n χρησιμοποιούμε τα παρακάτω:
- http://factordb.com/ : Database με factorings αριθμών.
- https://www.alpertron.com.ar/ECM.HTM : Web app που κάνει factor με ECM και SIQS.

ΥΓ: ΙΣΩΣ Η ΑΡΧΗ ΤΗΣ ΠΑΡΟΥΣΙΑΣΗΣ ΠΕΡΙΕΧΕΙ ΚΑΠΟΙΟ ΚΡΥΦΟ ΜΗΝΥΜΑ ΓΙΑ ΤΟ CHALLENGE ;)