Project I
Συμμετρική Κρυπτογραφία

Ασημάκης Κύδρος
ΑΕΜ: 3881
asimakis@csd.auth.gr

9 Μαΐου 2023

Άσκηση 1:
Το μήνυμα *'οκηθμφδζθγοθχυκχσφθμφμχγ'* έχει κρυπτογραφηθεί με τον παρακάτω αλγόριθμο:

Κάθε γράμμα του αρχικού μηνύματος αντικαθίσταται με την αριθμητική του τιμή $(\alpha \to 1, \beta \to 2, ..., \omega \to 24)$***, η οποία μετέπειτα προσθέτεται με την τιμή*** $f(x_0)$ ***του πολυωνύμου*** $f(x) = x^5 + 3x^3 + 7x^2 + 3x^4 + 5x + 4$***, όπου*** x_0 ***μια ρίζα του πολυωνύμου*** $g(x) = x^2 + 3x + 1$***. Το αποτέλεσμα αντικαθίσταται με το πλέον αντίστοιχο γράμμα.***

Αποκρυπτογραφήστε το.

Λύνοντας το $g(x)$ με διακρίνουσα καταλήγουμε στις ρίζες

$$x_0 = -1 - \phi, \quad x_0' = -1 - \bar{\phi}$$

όπου ϕ η χρυσή τομή. Άρρητες τιμές άρα η προσέγγιση αυτή δεν είναι ιδανική.

Κάνουμε διαίρεση πολυωνύμων μεταξύ των $f(x), g(x)$:

$$\begin{array}{rl|l} x^5 + 3x^4 + 3x^3 + 7x^2 + 5x + 4 & & x^2 + 3x + 1 \\ \cline{3-3} -x^5 - 3x^4 - x^3 & & x^3 + 2x + 1 \\ \cline{1-1} 2x^3 + 7x^2 + 5x & & \\ -2x^3 - 6x^2 - 2x & & \\ \cline{1-1} x^2 + 3x + 4 & & \\ -x^2 - 3x - 1 & & \\ \cline{1-1} 3 & & \end{array}$$

Επομένως ισχύει γενικά πως $f(x) = (x^3 + 2x + 1)g(x) + 3$. Άρα $f(x_0) = 3$, δηλαδή η κρυπτογράφηση εν τέλη χρησιμοποιεί το **κρυπτοσύστημα του Καίσαρα**.

Για να αποκρυπτογραφήσουμε το μήνυμα αντικαθιστούμε κάθε γράμμα με το γράμμα τρεις θέσεις πριν από αυτό. Εφαρμόζοντάς το αυτό ανακαλύπτουμε ότι το αρχικό μήνυμα ήταν

$$\boldsymbol{\mu\eta\delta\epsilon\iota\sigma\ \alpha\gamma\epsilon\omega\mu\epsilon\tau\rho\eta\tau\sigma\sigma\ \epsilon\iota\sigma\iota\tau\omega}$$

το οποίο ήταν η επιγραφή στην είσοδο της Ακαδημίας του Πλάτωνα.

Στο task 1.py υλοποιείται ακριβώς η παραπάνω διαδικασία, και επιβεβαιώνει το αποτέλεσμα.

Άσκηση 2:
Αποκρυπτογραφήστε τα κείμενα που βρίσκονται στο github και κρυπτογραφήθηκαν με τον αλγόριθμο του Vigenère. Βρείτε το μήκος του κλειδιού εφαρμόζοντας test Kasiski ή μέθοδο Friedman.

Έστω το κάθε κρυπτοκείμενο είναι της μορφής $c_1 c_2 \cdots c_k$.

Βρίσκουμε το (πιθανό) μήκος του κάθε κλειδιού εφαρμόζοντας το τεστ του Friedman: ξεκινάμε υποθέτοντας μήκος $\rho = 2$ και τρέχοντας τις παρακάτω λέξεις

$$C_1 = [c_1, c_{1+\rho}, ...]$$
$$C_2 = [c_2, c_{2+\rho}, ...]$$
$$\vdots$$
$$C_\rho = [c_\rho, c_{2\rho}, ...]$$

υπό το τεστ

$$IC(C_i) \approx^{?} 0.0665$$

όπου

$$IC = \frac{\sum_{i=1}^{26} m_i^2}{k^2}$$

με m_i το πλήθος εμφανίσεων του i-οστού γράμματος του αλφαβήτου στην λέξη.

Επαναλαμβάνουμε το παραπάνω για όλα τα ρ μέχρι κάποιο να περάσει το τεστ ή να φτάσουμε το μήκος του κρυπτογράμματος (το κλειδί δεν θα είναι ποτέ τόσο μεγάλο).

Τρέχοντας το παραπάνω για τα δύο κρυπτομηνύματα παίρνουμε (πιθανό) μήκος κλειδιού 7 και για τα δύο. Έχοντας τα μήκη των κλειδιών, μπορούμε να τα προσεγγίσουμε. Κάνουμε συχνοτική ανάλυση σε κάθε κρυπτόγραμμα, χωρίζοντάς το σε 7 (όσα και το μήκος του κλειδιού) υπό-μηνύματα με τέτοιον τρόπο ώστε κάθε γράμμα του δωθέν υπο-μηνύματος να κρυπτογραφήθηκε με το ίδιο γράμμα του κλειδιού. Κατ΄ αυτόν τον τρόπο, τα υπό-μηνύματα είναι σαν να είναι κρυπτογράμματα κρυπτογράφησης του Καίσαρα.

Εφόσον για κάθε υπο-μήνυμα το κλειδί-κομμάτι είναι στανταρ, ψάχνουμε σε καθένα το πιο συχνό γράμμα. Στοιχιματίζουμε πως αυτό θα αντιστοιχεί σε κάποιο από τα πιο συχνά γράμματα του αγγλικού αλφαβήτου, δηλαδή στο **E** ή **T**. Δημιουργούμε μια λέξη με όλα αυτά τα γράμματα.

Αξιοποιούμε το παραπάνω στοίχημα αποκρυπτογραφόντας τις προαναφαιρόμενες 2 λέξεις με κάθε συνδυασμό στο $\{E, T\}^7$. Κάποιο από τα αποτελέσματα της αποκρυπτογράφησης είναι το κλειδί ή το προσεγγίζει αρκετά, και βλέποντάς τα μπορούμε πιθανώς να τα προσεγγίσουμε.

Για το πρώτο κείμενο είναι αρκετά προφανές ότι το κλειδί είναι η λέξη ***EMPEROR***, διότι υπάρχει στις περιπτώσεις, όλες οι υπόλοιπες το προσεγγίζουν και επίσης αποκρυπτογραφεί το κείμενο τέλεια.

Για το δεύτερο δεν είμαστε τόσο τυχεροί, αλλά παρατηρούμε πως οι λέξεις **SLANNON** και **SWANNON** που υπάρχουν στις περιπτώσεις παράγουν αποκρυπτογράφηση που βγάζει αρκετό νόημα, και όλες οι υπόλοιπες φαίνεται να τα προσεγγίζουν. Μπορούμε επομένως να μαντέψουμε, και από την μερική αποκρυπτογράφηση, ότι το δεύτερο κλειδί είναι η λέξη ***SHANNON***, και όντως αυτή αποκρυπτογραφεί το κείμενο τέλεια.

Η υλοποίηση του παραπάνω μπορεί να βρεθεί στο task 2.py και τα αποτελέσματα της αποκρυπτογράφησης στο task 2 results.txt. Το πρώτο είναι ο λόγος του Charlie Chaplin στο φινάλε του *The Great Dictator*. Το δεύτερο είναι απόσπασμα από τον λόγο *Creative Thinking* του Claude Shannon.

Άσκηση 3:
Έστω m ένα 16-bit μήνυμα που κρυπτογραφείται σε c με τον εξής τρόπο:

$$c = m \oplus (m << 6) \oplus (m << 10)$$

Βρείτε τον γενικό τύπο αποκωδικοποίησης και υλοποιείστε τον σε κώδικα.

Έστω ότι

$$m = M_0M_1M_2M_3M_4M_5M_6M_7M_8M_9M_{10}M_{11}M_{12}M_{13}M_{14}M_{15}$$
$$c = C_0C_1C_2C_3C_4C_5C_6C_7C_8C_9C_{10}C_{11}C_{12}C_{13}C_{14}C_{15}$$

όπου M, C bits. Ισχύουν τα παρακάτω:

m	$m << 6$	$m << 10$	c
M_0	M_6	M_{10}	$M_0 \oplus M_6 \oplus M_{10}$
M_1	M_7	M_{11}	$M_1 \oplus M_7 \oplus M_{11}$
M_2	M_8	M_{12}	$M_2 \oplus M_8 \oplus M_{12}$
M_3	M_9	M_{13}	$M_3 \oplus M_9 \oplus M_{13}$
M_4	M_{10}	M_{14}	$M_4 \oplus M_{10} \oplus M_{14}$
M_5	M_{11}	M_{15}	$M_5 \oplus M_{11} \oplus M_{15}$
M_6	M_{12}	M_0	$M_6 \oplus M_{12} \oplus M_0$
M_7	M_{13}	M_1	$M_7 \oplus M_{13} \oplus M_1$
M_8	M_{14}	M_2	$M_8 \oplus M_{14} \oplus M_2$
M_9	M_{15}	M_3	$M_9 \oplus M_{15} \oplus M_3$
M_{10}	M_0	M_4	$M_{10} \oplus M_0 \oplus M_4$
M_{11}	M_1	M_5	$M_{11} \oplus M_1 \oplus M_5$
M_{12}	M_2	M_6	$M_{12} \oplus M_2 \oplus M_6$
M_{13}	M_3	M_7	$M_{13} \oplus M_3 \oplus M_7$
M_{14}	M_4	M_8	$M_{14} \oplus M_4 \oplus M_8$
M_{15}	M_5	M_9	$M_{15} \oplus M_5 \oplus M_9$

Παρατηρούμε πως το κρυπτόγραμμα c κουβαλάει το ίδιο όλη την απαραίτητη πληροφορία για την αποκρυπτογράφησή του. Παρατηρούμε επίσης πώς

$$M_6 = C_2 \oplus C_4 \oplus C_6 \oplus C_8 \oplus C_{10}$$
$$M_5 = C_1 \oplus C_3 \oplus C_5 \oplus C_7 \oplus C_9$$

Με το μάτι επιβεβαιώνουμε πως η παραπάνω σχέση γενικεύεται, αν σκεφτούμε το bitstream του κρυπτογράμματος ως 'κορδέλα'.

Επομένως συνεπάγουμε πως:

$$m = (M_i : i \in [0, 15]) \ni$$
$$M_i = C_{i-_{16}4} \oplus C_{i-_{16}2} \oplus C_i \oplus C_{i+_{16}2} \oplus C_{i+_{16}4}$$

όπου $-_{16}, +_{16}$ οι αντίστοιχες πράξεις $mod16$.

Η παραπάνω διαδικασία υλοποιείται ολόιδια στο task 3.py. Τρέχοντας το τεστ της main βλέπουμε πως το original plaintext και το αποτέλεσμα της αποκρυπτογράφησης ταυτίζονται.

Άσκηση 4:
Να αποδείξετε ότι, αν στο σύστημα μετατόπισης διαλέγουμε τυχαία τα κλειδιά από το σύνολο $\{0, 1, ..., 23\}$, τότε το σύστημα έχει τέλεια ασφάλεια.

Όντως, αν για κάθε γράμμα του plaintext διαλέγουμε ένα κλειδί ομοιόμορφα στην τύχη τότε το σύστημα γίνεται τελείως ασφαλές, και αυτό φαίνεται από 2 πράγματα:

1. αναλύσεις κρυπτογραμμάτων με οποιαδήποτε μετρική, όπως συχνότητα γραμμάτων, χάνουν την ουσία τους, και
2. το συνολικό κλειδί θα έχει μήκος ίσο με το plaintext, και άρα μέσω του θεωρήματος του Shannon το σύστημα πρέπει να έχει τέλεια ασφάλεια.

Άσκηση 5:
Υλοποιήστε τον OTP, μετατρέποντας εσωτερικώς το δωσμένο plaintext σε bits βάσει του παρακάτω πίνακα. Να δουλεύει και η κρυπτογράφηση και η αποκρυπτογράφηση και κάθε αποτέλεσμα να είναι σε text, όχι bits.

(0) A : 00000	(16) Q : 10000
(1) B : 00001	(17) R : 10001
(2) C : 00010	(18) S : 10010
(3) D : 00011	(19) T : 10011
(4) E : 00100	(20) U : 10100
(5) F : 00101	(21) V : 10101
(6) G : 00110	(22) W : 10110
(7) H : 00111	(23) X : 10111
(8) I : 01000	(24) Y : 11000
(9) J : 01001	(25) Z : 11001
(10) K : 01010	(26) . : 11010
(11) L : 01011	(27) ! : 11011
(12) M : 01100	(28) ? : 11100
(13) N : 01101	(29) (: 11101
(14) O : 01110	(30)) : 11110
(15) P : 01111	(31) - : 11111

Υλοποιούμε το Σημειωματάριο Μιας Φοράς όπως μας καθοδηγεί η θεωρία: η συνάρτηση δέχεται το plaintext σε μορφή κειμένου, και ενδεχομένως ένα κλειδί σε bits. Μετατρέπουμε το κείμενο-μήνυμα στα bits του για να ξέρουμε το πλήθος τους και προχωράμε στο κλειδί. Αν δεν δώθηκε ένα τότε δημιουργούμε ένα άλλο (ψεύδο)τυχαία.

Εκτελούμε bitwise xor το κλειδί με τα bits του plaintext και, βάσει του ανιστρόφου του αρχικού πίνακα, μετατρέπουμε το αποτέλεσμα σε κείμενο. Επιστρέφουμε το κλειδί και το κρυπτόγραμμα.

Τρέχοντας το test της main επιβεβαιωνόμαστε πως η υλοποίηση είναι σωστή. Προφανώς, ο OTP είναι και ο ίδιος του ο decryptor. Απλά απαιτείται να δωθεί το σωστό κλειδί.

Η υλοποίηση δίνεται από το task 5.py.

Άσκηση 6:
Βρείτε το πλήθος των ανάγωγων πολυωνύμων βαθμού 10 στο σώμα $\mathbb{F}_2$, αν αυτό δίνεται από τον εξής τύπο

$$N_2(n) = \frac{1}{n}\sum_{d|n}\mu(d)2^{n/d}, \quad n: \quad degree$$

όπου

$$\mu(d) = \begin{cases} 1, & d = 1 \\ (-1)^k, & d = p_1 p_2 .... p_k \quad (p_i \quad primes) \\ 0, & otherwise \end{cases}$$

Έχουμε

$$N_2(10) = \frac{1}{10}\sum_{d|10}\mu(d)2^{10/d} \tag{1}$$

Οι διαιρέτες του 10 είναι $\{1,\ 2,\ 5,\ 10\}$. Επομένως η (1) γίνεται

$$N_2(10) = \frac{1}{10}(\mu(1)2^{10} + \mu(2)2^5 + \mu(5)2^2 + \mu(10)2) \tag{2}$$

Εξ΄ ορισμού της μ ισχύουν τα εξής:

$$\begin{aligned} \mu(1) &= +1 \\ 2 \ \ prime \Rightarrow k = 1 &\Rightarrow \mu(2) = -1 \\ 5 \ \ prime \Rightarrow k = 1 &\Rightarrow \mu(5) = -1 \\ 10 = 2 \cdot 5 \Rightarrow k = 2 &\Rightarrow \mu(10) = +1 \end{aligned}$$

Βάσει των παραπάνω, η (2) γίνεται

$$N_2(10) = \frac{1}{10}(2^{10} - 2^5 - 2^2 + 2) = \frac{1}{10} \cdot 990 = 99$$

Η παραπάνω λύση υλοποιείται ολόιδια στο task 6.py αλλά και στο task 6.sage, όπως ζητείται.
Για την εύρεση των πρώτων παραγόντων χρησιμοποιείται η trial_division των σημειώσεων.

Άσκηση 7:
Υλοποιήστε τον RC4 και, χρησιμοποιώντας το κλειδί *'HOUSE'* κρυπτογραφήστε το μήνυμα *'MISTAKESAREASSERIOUSASTHERESULTSTHEYCAUSE'* Η υλοποίηση να αποκρυπτογραφεί σωστά. Χρησιμοποιήστε τον πίνακα της άσκησης 5.

Δεχόμαστε το plaintext και το κλειδί σε μορφή κειμένου. Μετατρέπουμε και τα δύο σε ακολουθίες ακεραίων με βάση τον δεδομένο πίνακα και περνάμε ως seed στον scheduler τη μετάφραση του κλειδιού.

Η υλοποίηση του δρομολογητή και της κλειδορροής ακολουθούν πιστά τη θεωρία, με μόνες αλλαγές ότι η κλειδορροή επιστρέφεται ολάκερη, όχι κομμάτι-κομμάτι, και κάθε κομμάτι είναι των 5 bit για να ταιρίαζει με τον δωσμένο πίνακα μετάφρασης.

Τέλος εκτελούμε byte-wise xor την κλειδορροή με τους κωδικούς-μετάφραση του plaintext και μετατρέπουμε το αποτέλεσμα σε κείμενο για να σχηματίσουμε το ciphertext.

Τρέχοντας το test της main επιβεβαιωνόμαστε πως και η κρυπτογράφηση και η αποκρυπτογράφηση έγιναν σωστά.
Παρατηρούμε πως το κλειδί 'HOUSE' επιστρέφει πάντα την ίδια κρυπτογράφηση, καθώς και πως ο RC4 αποκρυπτογραφεί τον εαυτό του με το σωστό κλειδί.
Και τα δύο αναμενόμενα καθώς αποτελεί ντετερμινιστική παραλλαγή του one time pad.

Η υλοποίηση δίνεται από το task 7.py.

Άσκηση 8:
Υπολογίστε την διαφορική ομοιομορφία (differential uniformity):

$$Diff(S) = \max_{\mathbf{x}\in\{0,1\}^6-\{\mathbf{0}\},\mathbf{y}\in\{0,1\}^4} |\{\mathbf{z}\in\{0,1\}^6 \ni S(\mathbf{z}\oplus\mathbf{x})\oplus S(\mathbf{z}) = \mathbf{y}\}|$$

του S-box S:

$$\begin{bmatrix} 0 & 2 & 3 & 7 & 9 & 12 & 15 & 7 & 6 & 15 & 15 & 1 & 7 & 3 & 1 & 0 \\ 1 & 5 & 6 & 13 & 4 & 1 & 5 & 11 & 7 & 8 & 7 & 1 & 1 & 3 & 2 & 13 \\ 5 & 3 & 5 & 12 & 11 & 1 & 1 & 5 & 13 & 0 & 15 & 7 & 2 & 2 & 13 & 0 \\ 3 & 12 & 3 & 11 & 2 & 2 & 2 & 4 & 6 & 5 & 5 & 0 & 4 & 3 & 1 & 0 \end{bmatrix}$$

Η λύση δίνεται από το task 8.py. Η θεωρία ορίζει πως η S τιμή του 6-bit αριθμού $X_0X_1X_2X_3X_4X_5$ δίνεται από την 4-bit τιμή που βρίσκεται στο κιβώτιο στη θέση

$$row = (X_0X_5)_{10}, \;\; col = (X_1X_2X_3X_4)_{10}$$

Η συνάρτηση coords υλοποιεί ακριβώς αυτό και επιστρέφει τις παραπάνω τιμές.

Έχοντας τις συντεταγμένες των $\mathbf{z}$, $\mathbf{z}\oplus\mathbf{x}$, κάνουμε bitwise xor τις αντίστοιχες τιμές και συγκρίνουμε το αποτέλεσμα με το $\mathbf{y}$. Αποθηκεύουμε το πλήθος των $\mathbf{z}$ που ικανοποιούν την παραπάνω ισότητα στη μεταβλητή **norm**.

Τρέχουμε την παραπάνω διαδικασία για όλα τα πιθανά $\mathbf{z}$ κόντρα σε κάθε πιθανό ζεύγος $(\mathbf{x}, \mathbf{y})$ του $\{0,1\}^6-\{000000\}\times\{0,1\}^4$. Η μέγιστη τιμή όλων των **norm** είναι και η ζητούμενη λύση. Για το δεδομένο κιβώτιο βγαίνει πως

$$Diff(S) = 14$$

Άσκηση 9:
Εξετάστε αν ισχύει το avalanche effect στον AES-128 φτιάχνοντας πολλά (>30) ζευγάρια μηνυμάτων που διαφέρουν σε ένα bit και μετρώντας σε πόσα bits διαφέρουν τα αντίστοιχα κρυπτομηνύματα. Δοκιμάστε με δύο καταστάσεις λειτουργίας (ECB και CBC) και χρησιμοποιώντας μήκος μηνυμάτων 2 block (δηλαδή 256-bit).

Η λύση δίνεται από το task 9.py. Για τη σωστή λειτουργία του προγράμματος απαιτείται το πακέτο pycryptodome το οποίο μπορεί να εγκατασταθεί επικαλώντας το pip install pycryptodome στη γραμμή εντολών. Υπάρχει περίπτωση η εγκατάσταση να κάνει conflict με το παλιό και πλέον deprecated πακέτο PyCrypto, σε περίπτωση που αυτό είναι ήδη εγκαταστημένο. Σε αυτή τη περίπτωση πρέπει πρώτα να απεγκατασταθεί το PyCrypto επικαλώντας το pip uninstall PyCrypto.

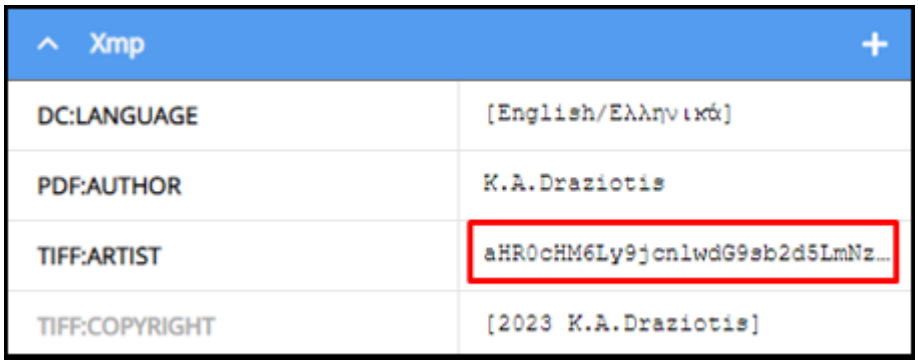
Αρχικά δημιουργούμε 100 μηνύματα, χρησιμοποιώντας κάθε φορά το τελευταίο μήνυμα της λίστας και αλλάζοντας ένα τυχαίο bit από αυτό για να προκύψει ένα καινούριο μήνυμα. Το υλοποιούμε έτσι ώστε, στη λίστα, κάθε στοιχείο να έχει 'γείτονες' δύο αριθμούς που διαφέρουν από αυτό ακριβώς σε ένα bit, ώστε η περαιτέρω δουλειά να είναι ενστικτώδης.

Δημιουργούμε δύο ciphers, ένα για κάθε κατάσταση λειτουργίας, και τρέχουμε και για τα δύο το εξής τεστ: Κρυπτογραφούμε δύο γείτονες και κάνουμε xor τα αντίστοιχα κρυπτομηνύματα. Το πλήθος των άσσων στο αποτέλεσμα δείχνει σε πόσα bits διαφέρουν τα δύο κρυπτομηνύματα (εξ΄ ορισμού της xor). Αποθηκεύουμε το άθροισμα αυτών των πληθών για τα 100 ζεύγη και το επιστρέφουμε ως ποσοστό.

Παρατηρούμε πως, για κατάσταση λειτουργίας ECB, ο AES σκοράρει πολύ άσχημα· είναι γνωστό πως το ECB-mode δεν πρέπει να χρησιμοποιείται για πολλούς λόγους. Αντιθέτως, για CBC ο AES σκοράρει κοντά στο 50%, επομένως ισχύει το avalanche effect.

Άσκηση 10:
Ανοίξτε το secure.zip με τη βοήθεια του εισαγωγικού pdf.

Περνάμε το pdf από έναν online pdf metadata extractor, εγώ χρησιμοποίησα αυτόν. Στα metadata θα βρούμε ένα string που μοίαζει να είναι κωδικοποιημένο σε Base64:



Το string αυτό είναι το

aHR0cHM6Ly9jcnlwdG9sb2d5LmNzZC5hdXRoLmdyOjgwODAvaG9tZS9wdWIvMTUv

και η αποκωδικοποίησή του από Base64, οποία έγινε με το εξής online tool, δίνει το παρακάτω link:

https://cryptology.csd.auth.gr:8080/home/pub/15/

Αυτό το link μας οδηγεί σε ένα custom sage site, το οποίο μας πληροφορεί ότι ο Walter White βρήκε το εξής μήνυμα στο εργαστήριό του, μια συνηθισμένη μέρα παραγωγής μπλε μεθαμφεταμίνης:

#2-75-22-6!

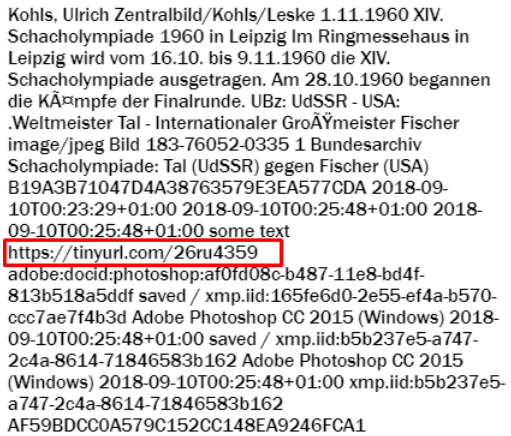
Ο Walter White ήταν χημικός, επομένως κάτι έχει να κάνει με χημεία το παραπάνω. Θεωρώντας τα νούμερα ατομικούς αριθμούς, παίρνουμε τα εξής στοιχεία:

$$2 \rightarrow \text{Helium (He)}$$
$$75 \rightarrow \text{Rhenium (Re)}$$
$$22 \rightarrow \text{Titanium (Ti)}$$
$$6 \rightarrow \text{Carbon (C)}$$

άρα το password είναι κάποια εκδοχή του #HeReTiC!, συγκεκριμένα το

#heretic!

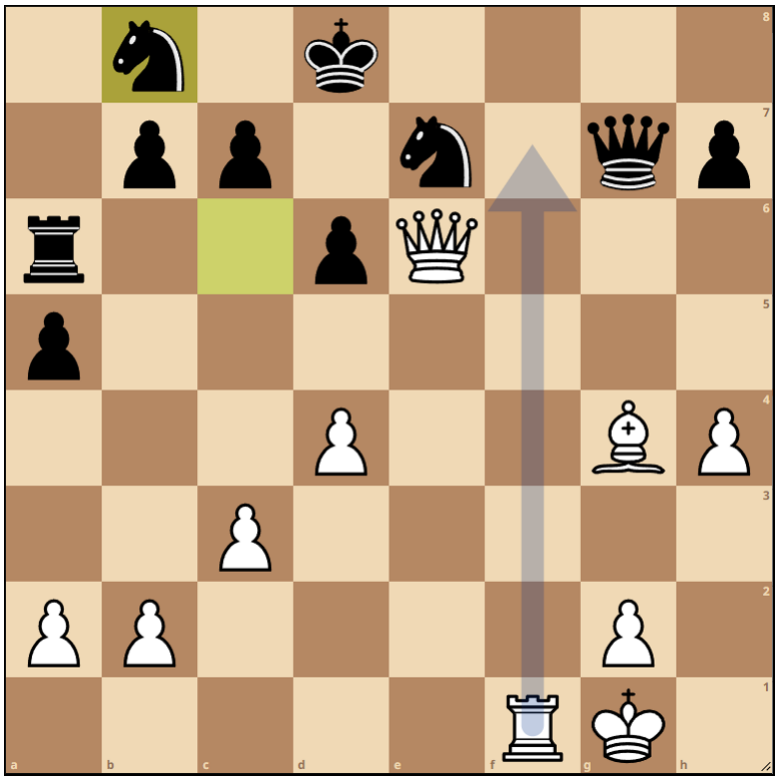
Το άνοιγμα του secret.txt μας οδηγεί στην εύρεση του κωδικού του secret2.zip. Το block που μας δίνεται ως hint είναι ξανά κωδικοποιημένο σε Base64, και η αποκωδικοποίησή του μας δίνει το decoded_hint.txt. Μέσα σε αυτό, μεταξύ άλλων, υπάρχει και ένα κομμάτι HTML κώδικα. Τρέχοντάς τον χρησιμοποιώντας κάποιο online tool οδηγούμαστε στο εξής wall of text:


Kohls, Ulrich Zentralbild/Kohls/Leske 1.11.1960 XIV. Schacholympiade 1960 in Leipzig Im Ringmessehaus in Leipzig wird vom 16.10. bis 9.11.1960 die XIV. Schacholympiade ausgetragen. Am 28.10.1960 begannen die KÃ¤mpfe der Finalrunde. UBz: UdSSR - USA: .Weltmeister Tal - Internationaler GroÃŸmeister Fischer image/jpeg Bild 183-76052-0335 1 Bundesarchiv Schacholympiade: Tal (UdSSR) gegen Fischer (USA) B19A3B71047D4A38763579E3EA577CDA 2018-09-10T00:23:29+01:00 2018-09-10T00:25:48+01:00 2018-09-10T00:25:48+01:00 some text https://tinyurl.com/26ru4359 adobe:docid:photoshop:af0fd08c-b487-11e8-bd4f-813b518a5ddf saved / xmp.iid:165fe6d0-2e55-ef4a-b570-ccc7ae7f4b3d Adobe Photoshop CC 2015 (Windows) 2018-09-10T00:25:48+01:00 saved / xmp.iid:b5b237e5-a747-2c4a-8614-71846583b162 Adobe Photoshop CC 2015 (Windows) 2018-09-10T00:25:48+01:00 xmp.iid:b5b237e5-a747-2c4a-8614-71846583b162 AF59BDCC0A579C152CC148EA9246FCA1

Αυτό το κείμενο αφορά την παρακάτω φωτογραφία, που βγήκε στην Σκακιστική Ολυμπιάδα της Λειψίας του 1960 μεταξύ των Bobby Fischer και Mikhail Tal:

(photo by German photographer Ulrich Kohls)

Εμάς μας ενδιαφέρει το link που φαίνεται υπογραμμισμένο, το οποίο μας οδηγεί σε ένα καινούριο custom sage site που μας προκαλεί να κερδίσουμε την παρακάτω παρτίδα σκάκι ως λευκά:



Το stockfish 15 δίνει την κίνηση που φαίνεται ως την καλύτερη, κοινώς την **Rf7**. Το site μας καθοδηγεί να την κάνουμε hash με MD5 για να ανοίξουμε το zip, και όντως το

f1f5e44313a1b684f1f7e8eddec4fcb0

λειτουργεί ως κλειδί και αποκαλύπτει το πιστοποιητικό-hash:

be121740bf988b2225a313fa1f107ca1