*Αν. Καθηγητής Π. Λουρίδας*

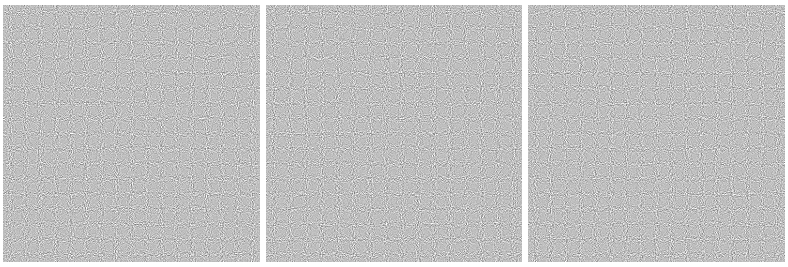
*Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας*

*Οικονομικό Πανεπιστήμιο Αθηνών*

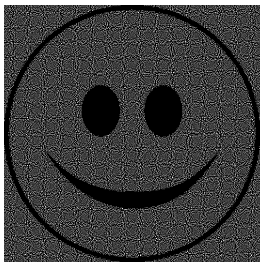
# Οπτική Κρυπτογραφία

Τρεις άνθρωποι, σε διαφορετικά μέρη του κόσμου, λαμβάνουν έναν φάκελο με μία εικόνα τυπωμένη σε μια διαφάνεια. Η εικόνα δεν δείχνει κάτι συγκεκριμένο· μοιάζει με μαύρες κουκίδες ατάκτως ερριμμένες στη διαφάνεια. Οι άνθρωποι αυτοί έχουν κανονίσει να συναντηθούν όλοι μαζί κάπου. Όταν μαζευτούν όλοι, τοποθετούν τις διαφάνειες που έχουν λάβει τη μία πάνω στην άλλη. Τότε, και μόνο τότε, οι διαφάνειες δημιουργούν μια εικόνα που περιέχει ένα μύνημα το οποίο μπορούν όλοι να διαβάσουν.

Αν αυτό μοιάζει με κινηματογραφικό σενάριο, δείτε τις παρακάτω τρεις εικόνες:



Αν τυπώσετε (μεγεθυμένες, ώστε να μη χαθεί η ανάλυσή τους) τις τρεις εικόνες σε διαφάνειες και τοποθετήσετε και τις τρεις τη μία πάνω στην άλλη, τότε θα εμφανιστεί η παρακάτω εικόνα:



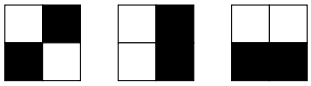
Αυτό το πετυχαίνουμε με την *οπτική κρυπτογραφία* (visual cryptography), στην οποία κρυπτογραφούμε το μήνυμά μας ώστε η αποκρυπτογράφησή του να είναι μια εικόνα με το μήνυμα ως περιεχόμενό της. Το παράδειγμά μας είναι επιπλέον μια εφαρμογή *διαμοιρασμού μυστικού* (secret sharing), όπου ένα μυστικό μύνημα διαμοιράζεται στα μέλη μιας ομάδας ώστε μόνο αν προκαθορισμένος αριθμός των μελών συνεισφέρουν τα μερίδια τους μπορεί να αποκαλυφθεί το μυστικό. Εδώ η ομάδα αποτελείται από τρία μέλη, και χρειάζονται και τα τρία μερίδια. Γενικότερα στο διαμοιρασμό μυστικού μπορούμε να έχουμε $n$ μέλη και να απαιτούνται $k \leq n$ μερίδια.

Ας δούμε πώς μπορούμε να διαμοιράσουμε ένα μυστικό με την οπτική κρυπτογραφία. Το μυστικό το οποίο θέλουμε να διαμοιράσουμε είναι το ίδιο μια εικόνα, η οποία αποτελείται από λευκά και μαύρα pixels. Στο παράδειγμά μας, το αρχικό μυστικό, το οποίο θέλουμε να διαμοιράσουμε ώστε να μην μπορεί να φανερωθεί εκτός και αν συνδυαστούν όλα τα μερίδια, είναι η γνωστή γελαστή φατσούλα:

Έστω λοιπόν ότι θέλουμε να την κρυπτογραφήσουμε σε τρία μερίδια. Παίρνουμε ένα-ένα pixel με τη σειρά, από τα αριστερά προς τα δεξιά, και από πάνω προς τα κάτω. Θα δημιουργήσουμε τρεις εικόνες-μερίδια, όπου στην κάθε μία θα αντικαταστήσουμε το pixel που εξετάζουμε στην αρχικής μας εικόνα με έναν πίνακα από pixels. Αν το pixel που εξετάζουμε είναι λευκό, θα το αντικαταστήσουμε από έναν πίνακα από pixels, εκ των οποίων ένας συγκεκριμένος αριθμός $b$ θα είναι μαύρα, και τα υπόλοιπα θα είναι λευκά. Αν το pixel που εξετάζουμε είναι λευκό, θα το αντικαταστήσουμε από έναν άλλο πίνακα pixels, εκ των οποίων ένας συγκεκριμένος αριθμός $b' < b$ θα είναι μαύρα, και τα υπόλοιπα θα είναι λευκά.

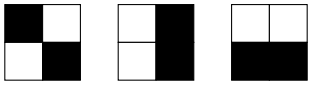
Με τον τρόπο αυτό θα προκύψουν τρία μερίδια. Αν τοποθετήσουμε τα μερίδια το ένα πάνω στο άλλο, οι περιοχές που αντιστοιχούν σε λευκό pixel της αρχικής εικόνας, θα έχουν λιγότερα μαύρα pixel από τις περιοχές που αντιστοιχούν σε μαύρο pixel της αρχικής εικόνας. Έτσι, θα προκύψει η γελαστή φατσούλα, όχι πλέον μαυρόασπρη μεν, αλλά μπορούμε να τη διακρίνουμε από τη διαφορά της φωτεινότητας μέσα στην εικόνα.

Συγκεκριμένα, θα μπορούσαμε να αντικαταστήσουμε ένα λευκό pixel με κάθε έναν από τους τρεις πίνακες που ακολουθούν (αφού θέλουμε τρία μερίδια):



Τότε, αν τοποθετήσουμε τους τρεις αυτούς πίνακες τον ένα πάνω στον άλλο (όπως θα κάναμε αν τοποθετούσαμε τα τρία μερίδια το ένα πάνω από το άλλο), θα πέρναμε:

Ομοίως, θα μπορούσαμε να αντικαταστήσουμε ένα μαύρο pixel με κάθε έναν από τους τρεις πίνακες που ακολουθούν:



Τότε, αν τοποθετήσουμε τους τρεις αυτούς πίνακες τον ένα πάνω στον άλλο (όπως θα κάναμε αν τοποθετούσαμε τα τρία μερίδια το ένα πάνω από το άλλο), θα πέρναμε:

Ενώ από την αλληλοεπικάλυψη των πινάκων για το λευκό pixel παίρνουμε έναν πίνακα με ένα λευκά pixel, από την αλληλοεπικάλυψη των πινάκων για το μαύριο pixel παίρνουμε έναν πίνακα εντελώς μαύρο. Έτσι, όταν συνδυαστούν τα τρία μερίδια, οι περιοχές της εικόνας που θα παραχθεί θα είναι λευκότερες (κατά 1/4) όταν αντιστοιχούν σε λευκό pixel, από τις περιοχές της εικόνας που αντιστοιχούν σε μαύρο pixel. Συνεπώς, η εικόνα που θα παραχθεί θα περιέχει, γκριζαρισμένη, την αρχική εικόνα, όπως είδαμε ότι πράγματι συμβαίνει.

Χρειαζόμαστε έναν γενικό τρόπο για να παράγουμε πίνακες με αυτά τα χαρακτηριστικά. Εμείς θα δούμε δύο κατασκευές, οι οποίοι μπορούν να χρησιμοποιηθούν για το διαμοιρασμό $k$ μυστικών σε $k$ μερίδια.

**Κατασκευή 1**

Ξεκινάμε με την δεύτερη κατασκευή. Φτιάχνουμε δύο σύνολα $k$ διανυσμάτων, τα $J_0^0$, $J_1^0$, ..., $J_{k-1}^0$ και $J_0^1$, $J_1^1$, ..., $J_{k-1}^1$. Κάθε ένα από τα διανύσματα έχει μήκος $k$. Τα διανύσματα $J_i^0$ κατασκευάζονται ως εξής: $J_i^0 = 0^i 10^{k-i}$ για $0 \leq k < k-1$ και $J_{k-1}^0 = 1^{k-1}0$. Άρα, για $k = 4$, είναι:

$$J_0^0 = [1, 0, 0, 0]$$
$$J_1^0 = [0, 1, 0, 0]$$
$$J_2^0 = [0, 0, 1, 0]$$
$$J_3^0 = [1, 1, 1, 0]$$

Τα διανύσματα $J_i^1$ κατασκευάζονται ως εξής: $J_i^1 = 0^i 10^{k-i}$ για $0 \leq k \leq k-1$. Άρα, για $k = 4$, είναι:

$$J_0^1 = [1, 0, 0, 0]$$
$$J_1^1 = [0, 1, 0, 0]$$
$$J_2^1 = [0, 0, 1, 0]$$
$$J_3^1 = [0, 0, 0, 1]$$

Στη συνέχεια, παίρνουμε όλα τα $2^k$ δυνατά δυαδικά διανύσματα $X_i$ μήκους $k$, όπου $X_i$ είναι η δυαδική αναπαράσταση του αριθμού $i$. Για παράδειγμα, $X_5 = [0, 1, 0, 1]$. Δημιουργούμε τότε δύο πίνακες $S^0$ και $S^1$ διαστάσεων $k \times 2^k$, ως εξής. Για τον $S^t$, $t \in \{0, 1\}$, θα έχουμε $S^t[i, j] = \langle J_i^t, X_j \rangle$, όπου με $\langle u, v \rangle$ συμβολίζουμε το εσωτερικό γινόμενο δύο διανυσμάτων $u$ και $v$. Προσοχή όμως! Το εσωτερικό γινόμενο θα το υπολογίσουμε κάνοντας τις πράξεις modulo 2, δηλαδή:

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| × | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Μπορείτε να παρατηρήσετε ότι η πρόσθεση και ο πολλαπλασιασμός δυαδικών αριθμών modulo 2 είναι το ίδιο με το αποκλειστικό Ή (XOR) και το λογικό και (AND).

Έχοντας κατασκευάσει τους πίνακες $S^t$, μπορούμε να προχωρήσουμε στην κρυπτογράφηση και δημιουργία των μεριδίων:

- Για κάθε pixel στη θέση $(i, j)$ της αρχικής εικόνας, από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω:
    1. Αν το pixel είναι 0, δημιούργησε από τον πίνακα $S^0$ έναν πίνακα $S'$ αναδιατάσσοντας τυχαία τις στήλες του $S^0$. Αλλιώς, δημιούργησε από τον πίνακα $S^1$ έναν πίνακα $S'$ αναδιατάσσοντας τυχαία τις στήλες του $S^1$.
    2. Από κάθε μία από τις $k$ γραμμές του πίνακα $S'$ δημιούργησε έναν πίνακα διαστάσεων $w \times h$ όπου $w = 2^{\lfloor k/2 \rfloor + k \bmod 2}$ και $h = 2^{\lfloor k/2 \rfloor}$.
    3. Στο $k$ μερίδιο της εικόνας τοποθέτησε τον $k$ από τους παραπάνω πίνακες $w \times h$ στη θέση $(i \times h, j \times w)$.

Αν η αρχική εικόνα έχει $r$ γραμμές και $c$ στήλες, το κάθε μερίδιο θα έχει $r \times h$ γραμμές και $c \times w$ στήλες. Για τέσσερα μερίδια, $k = 4$, επομένως $w = h = 2^2$, συνεπώς το μέγεθος της εικόνα θα δεκαεξαπλασιαστεί στα μερίδια. Για τρία μερίδια, $k = 3$, επομένως $w = 4$ και $h = 2$, οπότε η γεωμετρία των μεριδίων θα είναι διαφορετική από τη γεωμετρία της αρχικής εικόνας.

Γενικότερα, στην πρώτη κατασκευή για την απεικόνιση κάθε pixel σε κάθε ένα από $k$ μερίδια χρησιμοποιούμε $2^k$ pixels (όσες οι στήλες των πινάκων $S^t$), και η φωτεινότητα των πινάκων που χρησιμοποιούμε για τα λευκά pixels διαφέρει από τη φωτεινότητα των πινάκων που χρησιμοποιούμε για τα μαύρα pixels κατά $1/2^k$.

### Κατασκευή 2

Στη δεύτερη κατασκευή, ξεκινάμε με ένα σύνολο $k$ στοιχείων, $W = \{e_0, e_1, \ldots, e_{k-1}\}$. Τα στοιχεία $e_i$ μπορεί να είναι οτιδήποτε, για παράδειγμα οι αριθμοί από το 0 έως και το $k - 1$, ή ό,τι άλλο μας βολεύει. Δημιουργούμε όλα τα δυνατά υποσύνολα του $W$ με άρτιο πλήθος στοιχείων, $\pi_0, \pi_1, \ldots, \pi_{2^k-2}$ και όλα τα δυνατά υποσύνολα του $W$ με περιττό πλήθος στοιχείων, $\sigma_0, \sigma_1, \ldots, \sigma_{2^k-2}$. Δεν έχει σημασία ποιο ακριβώς είναι το κάθε ένα από τα $\pi_i, \sigma_i$, δηλαδή η σειρά με την οποία τα παράγουμε. Μια μέθοδος είναι, αφού έχουμε $k$ στοιχεία, πάρουμε τα υποσύνολα θεωρώντας όλους τους δυαδικούς αριθμούς με $k$ ψηφία. Για παράδειγμα, αν $k = 3$, ο 000 είναι το κενό σύνολο και το αντιστοιχούμε στο $\pi_0$, ο 001 είναι το υποσύνολο που περιέχει τον πρώτο αριθμό από το $W$ και το αντιστοιχούμε στο $\sigma_0$, ο 010 είναι το υποσύνολο που περιέχει τον δεύτερο αριθμό από το $W$ και το αντιστοιχούμε στο $\sigma_1$, ο ο 011 είναι το υποσύνολο που περιέχει τους πρώτους δύο αριθμούς από το $W$ και το αντιστοιχούμε στο $\pi_1$, κ.ο.κ.

Με αυτά κατασκευάζουμε δύο πίνακες $S^0$ και $S^1$ διαστάσεων $k \times 2^{k-1}$ ως εξής: $S^0[i,j] = 1$ αν και μόνο αν $e_i \in \pi_j$ και $S^1[i,j] = 1$ αν και μόνο αν $e_i \in \sigma_j$. Στη συνέχεια, η παραγωγή των μεριδίων είναι σχεδόν η ίδια με την πρώτη κατασκευή:

- Για κάθε pixel στη θέση $(i,j)$ της αρχικής εικόνας, από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω:
  1. Αν το pixel είναι 0, δημιούργησε από τον πίνακα $S^0$ έναν πίνακα $S'$ αναδιατάσσοντας τυχαία τις στήλες του $S^0$. Αλλιώς, δημιούργησε από τον πίνακα $S^1$ έναν πίνακα $S'$ αναδιατάσσοντας τυχαία τις στήλες του $S^1$.
  2. Από κάθε μία από τις $k$ γραμμές του πίνακα $S'$ δημιούργησε έναν πίνακα διαστάσεων $w \times h$ όπου $w = 2^{\lfloor (k-1)/2 \rfloor + (k-1) \bmod 2}$ και $h = 2^{\lfloor (k-1)/2 \rfloor}$.
  3. Στο $k$ μερίδιο της εικόνας τοποθέτησε τον $k$ από τους παραπάνω πίνακες $w \times h$ στη θέση $(i \times h, j \times w)$.

Βρήκατε τη διαφορά; Είναι στο βήμα 2, όπου οι διαστάσεις $w$ και $h$ είναι διαφορετικές, μικρότερες από πριν, αφού οι πίνακες $S^0$ και $S^1$ έχουν $2^{k-1}$ αντί για $2^k$ στήλες. Πράγματι, η κατασκευή αυτή είναι λίγο καλύτερη από την πρώτη κατασκευή. Για την απεικόνιση κάθε pixel σε κάθε ένα από $k$ μερίδια χρησιμοποιούμε $2^{k-1}$ pixels, και η φωτεινότητα των πινάκων που χρησιμοποιούμε για τα λευκά pixels διαφέρει από τη φωτεινότητα των πινάκων που χρησιμοποιούμε για τα μαύρα pixels κατά $1/2^{k-1}$.

Σκοπός της εργασίας είναι να φτιάξετε ένα πρόγραμμα σε Python που να υλοποιεί αυτές τις δύο κατασκευές και την αποκρυπτογράφηση των μεριδίων.

## Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*
- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2023-4`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `visual_cryptography.py`.
- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.

- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση των παρακάτω βιβλιοθηκών ή τμημάτων τους όπως ορίζεται:
  - `sys.argv`
  - `argparse`
  - `permutations`
  - `seed`
  - `sample`
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.
- Η έξοδος του προγράμματος θα πρέπει να περιλαμβάνει μόνο ό,τι φαίνεται στα παραδείγματα. *Η φλυαρία δεν επιβραβεύεται.*

Στο βήμα 1 των κατασκευών παίρνουμε κάθε φορά τυχαία τις στήλες των πινάκων $S^0$ και $S^1$. Υπάρχουν δύο τρόποι που μπορούμε να το κάνουμε αυτό:

- Παίρνουμε όλες τις δυνατές διατάξεις των στηλών με τη χρήση της συνάρτησης `permutations` και επιλέγουμε κάθε φορά τυχαία μια από αυτές.
- Παίρνουμε κάθε φορά μια τυχία διάταξη των στηλών με τη χρήση της συνάρτησης `sample` και συγκεκριμένα με το ιδίωμα

  ```
  sample(x, k=len(x))
  ```

  το οποίο επιστρέφει τυχαία ανακατεμένα τα στοιχεία `x`.

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
visual_cryptography.py [-c {c1,c2}] [-s SHARES] [-d DECRYPT]
                       [-r RANDOM_SEED] [image ...]
```

Η σημασία των παραμέτρων είναι η εξής:

- Η παράμετρος `-c`, αν υπάρχει, καθορίζει αν θα χρησιμοποιηθεί η κατασκευή 1 ή η κατασκευή 2. Επιτρεπόμενες τιμές είναι η `c1`, για την κατασκευή 1, και `c2`, για την κατασκευή 2. Αν δεν υπάρχει η παράμετρος, το πρόγραμμα θα χρησιμοποιεί την κατασκευή 1.
- Η παράμετρος `-s` δίνει τον αριθμό των μεριδίων.
- Η παράμετρος `-d` σημαίνει ότι το πρόγραμμα θα κάνει αποκρυπτογράφηση. Αλλιώς θα κάνει κρυπτογράφηση. Η τιμή της παραμέτρου είναι το όνομα του αρχείου που θα περιέχει την αποκρυπτογραφημένη εικόνα.
- Η παράμετρος `-r`, αν υπάρχει, δίνει τον σπόρο (seed) με τον οποίο θα αρχικοποιείται η γεννήτρια ψευδοτυχαίων αριθμών της Python με τη συνάρτηση `seed`. Με τον τρόπο αυτό θα μπορείτε να δημιουργείτε την ίδια έξοδο, ντετερμινιστικά.

- Στο τέλος της γραμμής εντολών για την κλήση του προγράμματος θα υπάρχει είτε το όνομα του αρχείου που περιέχει την εικόνα προς κρυπτογράφηση, είτε τα μερίδια της εικόνας που θα χρησιμοποιηθούν για την αποκρυπτογράφηση. Αν το πρόγραμμα κάνει κρυπτογράφηση, κάθε μερίδιο θα έχει όνομα `enc_image_i_c1.txt` ή `enc_image_i_c2.txt` όπου `image` είναι το όνομα του αρχείου προς κρυπτογράφηση χωρίς κατάληξη, `i` είναι το όνομα του μεριδίου, και `c1` ή `c2` υποδεικνύει την κατασκευή που χρησιμοποιήθηκε. Τα αρχεία που θα χειρίζεται το πρόγραμμα θα είναι αρχεία κειμένου. Κάθε γραμμή θα περιέχει μια σειρά από 0 και 1, με 1 να αντιστοιχεί σε μαύρο και 0 να αντιστοιχεί σε λευκό. Δείτε παρακάτω στα παραδείγματα που δίνονται.

## Παραδείγματα

### Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python visual_cryptography.py -s 4 laughing_smiley_bw.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο `laughing_smiley_bw.txt` και θα παράξει τέσσερα αρχεία:
`enc_laughing_smiley_bw_0_c1.txt`
`enc_laughing_smiley_bw_1_c1.txt`
`enc_laughing_smiley_bw_2_c1.txt`
`enc_laughing_smiley_bw_3_c1.txt`.

Δεδομένου ότι τα αρχεία παράγονται τυχαία, τα δικά σας αρχεία δεν είναι απαραίτητο ότι θα έχουν τα ίδια περιεχόμενα με τα παραπάνω. Αλλά θα πρέπει να αποκρυπτογραφούνται σωστά.

### Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python visual_cryptography.py -d dec_laughing_smiley_bw_c1.txt
       enc_laughing_smiley_bw_0_c1.txt
       enc_laughing_smiley_bw_1_c1.txt
       enc_laughing_smiley_bw_2_c1.txt
       enc_laughing_smiley_bw_3_c1.txt
```

τότε το πρόγραμμα θα διαβάσει τα τέσσερα αρχεία:
`enc_laughing_smiley_bw_0_c1.txt`
`enc_laughing_smiley_bw_1_c1.txt`
`enc_laughing_smiley_bw_2_c1.txt`
`enc_laughing_smiley_bw_3_c1.txt`.

και θα παράξει το αρχείο `dec_laughing_smiley_bw_c1.txt`.

**Παράδειγμα 3**

Αν ο χρήστης του προγράμματος δώσει:

```
python visual_cryptography.py -s 3 laughing_smiley_bw.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο `laughing_smiley_bw.txt` και θα παράξει τρία αρχεία:
`enc_laughing_smiley_bw_0_c2.txt`
`enc_laughing_smiley_bw_1_c2.txt`
`enc_laughing_smiley_bw_2_c2.txt`.

Δεδομένου ότι τα αρχεία παράγονται τυχαία, τα δικά σας αρχεία δεν είναι απαραίτητο ότι θα έχουν τα ίδια περιεχόμενα με τα παραπάνω. Αλλά θα πρέπει να αποκρυπτογραφούνται σωστά.

**Παράδειγμα 4**

Αν ο χρήστης του προγράμματος δώσει:

```
python visual_cryptography.py -d dec_laughing_smiley_bw_c2.txt
       enc_laughing_smiley_bw_0_c2.txt
       enc_laughing_smiley_bw_1_c2.txt
       enc_laughing_smiley_bw_2_c2.txt
```

τότε το πρόγραμμα θα διαβάσει τα τρία αρχεία:
`enc_laughing_smiley_bw_0_c2.txt`
`enc_laughing_smiley_bw_1_c2.txt`
`enc_laughing_smiley_bw_2_c2.txt`.

και θα παράξει το αρχείο `dec_laughing_smiley_bw_c2.txt`.

## Περισσότερες Πληροφορίες

Η τεχνική οπτικής κρυπτογραφίας που είναι αντικείμενο αυτής της εργασίας είναι επινόηση των Moni Naor και Adi Shamir Naor και Shamir 1995.

# References

Naor, Moni, and Adi Shamir. 1995. “Visual cryptography”. In *Advances in Cryptology—EUROCRYPT’94,* edited by Alfredo De Santis, 1–12. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-44717-7.

Καλή Επιτυχία!