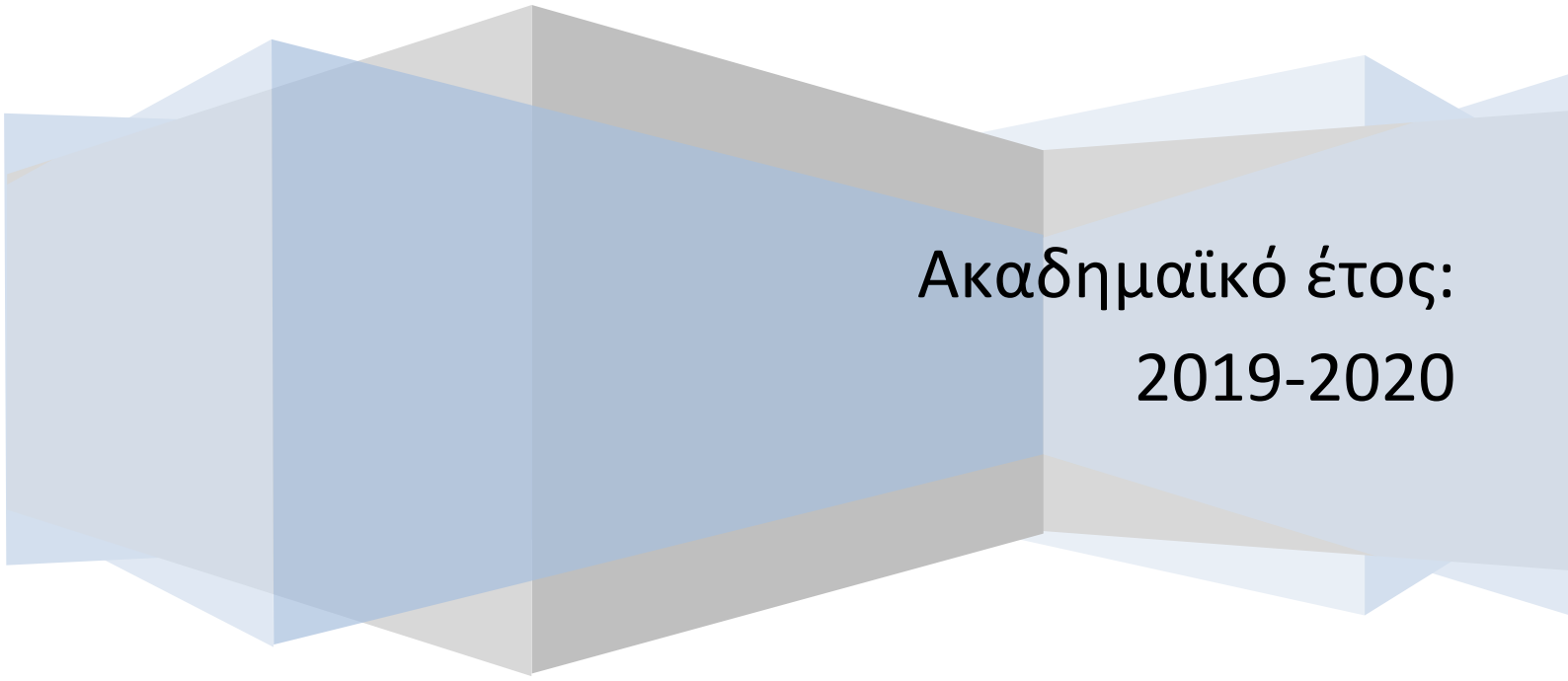
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Ασφάλεια Υπολογιστών και Δικτύων

Αναφορά εργασίας:
Υλοποίηση αλγορίθμου S-DES σε γλώσσα προγραμματισμού python με κατάλληλο GUI

ΒΑΖΑΙΟΣ ΣΤΥΛΙΑΝΟΣ Α.Μ.:1054284 CEID
email: vazaios@ceid.upatras.gr



Ακαδημαϊκό έτος:
2019-2020

Περιεχόμενα

Περίληψη

Η βασική λειτουργία του προγράμματος είναι να υλοποιήσει τον αλγόριθμο κρυπτογράφησης Simplified Data Encryption Standard (S-DES) οποίος βασίζεται στην κρυπτογράφηση μπλοκ. Παίρνει σαν είσοδο ένα μπλοκ 8-bit απλού κειμένου ή 8-bit και ένα κλειδί μεγέθους 10-bit και τελικά παράγει ένα κρυπτογραφημένο μπλοκ 8-bit cipher-text ως έξοδο. Πρόκειται για ένα συμμετρικό κρυπτογράφο που έχει δύο επαναλήψεις, ενώ για την κρυπτογράφηση\αποκρυπτογράφηση χρησιμοποιούνται συγκεκριμένες μεταθέσεις των bits που ορίζονται από τον ίδιο τον αλγόριθμο. Για τις μεταθέσεις αυτές υπάρχουν συγκεκριμένοι πίνακες που ορίζουν την σειρά με την οποία θα γίνονται: P10, P8, P4 EP(Expand&Permutate), και οι πίνακες S0-box και S1-box. Επίσης το πρόγραμμά μας διαθέτει την δυνατότητα αποθήκευσης των κρυπτογραφημένων και αποκρυπτογραφημένων bits\texts σε αρχεία .txt στους φακέλους Encrypted Documents και Decrypted Documents αντίστοιχα καθώς και ο χρήστης μπορεί να τοποθέτηση αρχεία .txt που θέλει να κρυπτογραφήσει μέσα στον φάκελο Documents στο directory του προγράμματος ώστε να γίνουν διαθέσιμα ως επιλογή στην είσοδο. Ακόμα πρέπει να τονιστεί ότι δίνεται η δυνατότητα εισόδου bits\ text και ολόκληρων αρχείων.

Οδηγίες Χρήσης:

Για την σωστή λειτουργία του προγράμματος συστήνεται τουλάχιστον η ανάλυση οθόνης fhd(1920x1080) ωστόσο λειτουργεί και σε μικρότερες αναλύσεις με την βοήθεια ενσωματωμένων scrollbars για την οριζόντια και κάθετη κατεύθυνση.

Ο python interpreter να είναι τουλάχιστον στην έκδοση 3.7 και να υπάρχουν τα εξής packages MouseInfo, Pillow, PyAutoGUI, PyGetWindow, PyMsgBox, PyRect, PyScreeze, pip, pyperclip και setuptools.

Για την έναρξη του προγράμματος εκτελέστε το αρχείο: S-Des with GUI.py

Έπειτα μπορείτε να προηγηθείτε στα μενού με τα κουμπιά Next, Back και Exit.

Στην δεύτερη σελίδα εισάγεται τον κλειδί των 10bit που επιθυμείτε και στην συνέχεια επιλέξτε μία από την ακόλουθες επιλογές με checkbox ανάλογα με την είσοδο που θέλετε. Έπειτα επικυρώστε την είσοδο σας και τελικά αποθηκεύστε τα bits και το κλειδί πατώντας το κουμπί “Save Key and Bits”. Πατώντας αυτό το κουμπί τρέχει στο παρασκήνιο ο αλγόριθμος S-DES και στις υπόλοιπες σελίδες που ακολουθούν θα εμφανιστεί η διαδικασία που ακολουθεί ο αλγόριθμος για ενδεικτικά τα πρώτα 8-bit της εισόδου. Σε κάθε μία από τις επόμενες τέσσερις σελίδες που ακολουθούν (δύο για κρυπτογράφηση και 2 για αποκρυπτογράφηση) επιλέξτε ένα ένα τα κουμπιά για την εμφάνιση του αποτελέσματος του κάθε βήματος. Μόλις τελειώσετε με τα βήματα της μίας σελίδας επιλέξτε την επόμενη μέχρι να φτάσετε στο τέλος.

Είσοδος:

Το πρόγραμμα αυτό έχει την δυνατότητα πολλαπλών εισόδων. Δηλαδή ο χρήστης μπορεί να δώσει τέσσερις διαφορετικούς τύπους εισόδου για να γίνει μετά με βάση αυτές η κρυπτογράφηση και αποκρυπτογράφηση S-DES. Αρχικά σαν είσοδο ο χρήστης θα πρέπει πρώτα να διαλέξει το κλειδί των 10-bit που επιθυμεί και στην συνέχεια να επιλέξει είτε να γράψει μία ακολουθία από bits είτε να γράψει αλφαριθμητικούς χαρακτήρες είτε να επιλέξει ένα ολόκληρο αρχείο που μπορεί να περιέχει είτε αλφαριθμητικούς χαρακτήρες είτε bits είτε και τα δυο μαζί. Πρέπει να τονιστεί πως όλα τα αλφαριθμητικά θα μετατραπούν με την βοήθεια της συνάρτηση strTobin σε bits ενώ σε περίπτωση που τα bits από τα αλφαριθμητικά είτε από τα bits εισόδου δεν μπορούν να ομαδοποιηθούν ακριβώς ανά 8bits θα προσθέσουν μηδενικά bits μέχρι να γίνει δυνατή αυτή η ομαδοποίηση. Αυτά τα προστιθέμενα bits θα αφαιρεθούν τελικά στην αποκρυπτογράφηση ώστε να έχουμε ακριβώς την είσοδο που είχαμε.

Έξοδος:

Το πρόγραμμα αυτό αποθηκεύει τα κρυπτογραφημένα bits στον φάκελο Encrypted Documents με συγκεκριμένη ονομασία αρχείου ανάλογα με την είσοδο. Συγκεκριμένα για την είσοδο που προέρχεται από αρχεία του φακέλου Documents αποθηκεύεται ως’’Encrypted_file_¨όνομα αρχικής εισόδου¨.txt”, για τα απλά αλφαριθμητικά και τις ακολουθίες των bits σαν είσοδο αποθηκεύονται "Encryption__text_string_¨ώρα κρυπτογράφησης¨.txt" και "Encryption_binary_¨ώρα κρυπτογράφησης¨.txt" αντίστοιχα Για τις αποκρυπτογραφημένες εξόδους χρησιμοποιείτε παρόμοια λογική. Αποθηκεύονται μέσα στον φάκελο “Decrypted Documents” και αντίστοιχα για τις εισόδους όπως και πριν απλά με την διαφορά ότι στην θέση του Encrypted θα είναι Decrypted.

Διαχείριση Σφαλμάτων:

Επίσης το πρόγραμμα μας έχει συναρτήσεις διαχείρισης Σφαλμάτων εισόδου που στην περίπτωση κάποιας λάθος εισόδου ή λάθος διαδικασίας εισόδου ή κάποιας απρόσμενης επιλογής του χρήστη θα εμφανίζεται στην οθόνη αντίστοιχο μήνυμα error με οδηγίες για την σωστή χρήση του προγράμματος και τι ακριβώς έγινε λάθος στην είσοδο. Παραδείγματα για αυτά τα σφάλματα μπορούν να βρεθούν παρακάτω στο κεφάλαιο Screenshots Προγράμματος.

Συναρτήσεις:

Το πρόγραμμα αυτό αποτελείτε συνολικά από 91 συναρτήσεις που συμβάλλουν στην ορθή και γρήγορη λειτουργία του. Καθεμία από αυτές τις συναρτήσεις παίζει μεγάλο ρόλο στην γενική ευστάθεια και στην διαχείριση των ενεργειών και τον δεδομένων του χρήστη.

renameSpaceToUnderscore() : Αλλάζει την ονομασία των αρχείων στον φάκελο Documents αντικαθιστώντας τα κενά με underscrore

strTobin(strtext): Μετατρέπει τους αλφαριθμητικούς χαρακτήρες σε δυαδικά ψηφία.

binTostr(bintext): Μετατρέπει τα δυαδικά ψηφία σε αλφαριθμητικούς χαρακτήρες.

addTill8bits(strtext,fillbits): Αν τα bits δεν μπορούν να ομαδοποιηθούν ακριβώς σε ομάδες των 8bits τότε προσθέτουμε μηδενικά bits μέχρι να μπορέσουμε.

save_text(entry): Αποθηκεύει την είσοδο είτε text είτε bits είτε αλφαριθμητικούς χαρακτήρες είτε ολόκληρα αρχεία και ανάλογα με τον τύπο της εισόδου τα μετατρέπει σε κατάλληλα bits για να χρησιμοποιούμουν παρακάτω.

inputError(): Σφάλματα που αφορούν το 10bit κλειδί.

fileError():Σφάλματα που αφορούν τα αρχεία στο Documents φάκελο.

no_inputError(): Σφάλματα που αφορούν την κενή εκτέλεση του προγράμματος.

unchecked_option(): Σφάλματα που αφορούν την μη επιλογή checkbox.

set_CheckBoxVar() : Δηλώνει την επιλογή checkbox

boxConfirmation(): Αν έχει επιλεχτεί κάποιο checkbox τότε μπορεί να δηλωθεί η σωστή επιλογή και να προχωρήσει στην εκτέλεση του SDES.

get_inputkey(Entry): Αποθήκευση του 10-bit κλειδιού.

raise_frame(frame): Επόμενο frame παραθύρου.

main_Encrypt_window(): Εκτέλεση του γραφικού περιβάλλοντος.

call_s1()-call_s14(): Αλλαγή της τιμής των μεταβλητών που εμφανίζονται με τις σωστές πλέον τιμές τους και όχι τις αρχικοποιημένες.

call_Es1()-call_Es14():Αλλαγή της τιμής των μεταβλητών που εμφανίζονται με τις σωστές πλέον τιμές τους και όχι τις αρχικοποιημένες.

call_E2s1()-call_E2s12(): Αλλαγή της τιμής των μεταβλητών που εμφανίζονται με τις σωστές πλέον τιμές τους και όχι της αρχικοποιημένες.

call_Des1()-call_Des14():Αλλαγή της τιμής των μεταβλητών που εμφανίζονται με τις σωστές πλέον τιμές τους και όχι της αρχικοποιημένες.

call_De2s1()-call_De2s12():Αλλαγή της τιμής των μεταβλητών που εμφανίζονται με τις σωστές πλέον τιμές τους και όχι της αρχικοποιημένες.

permute(Bitkey, permSequence): Μετάθεση των δυαδικών ψηφίων με βάση των Πινάκων που δίνονται

divPermKeys(key): Διαχωρισμός μιας ακολουθίας από bits στην μέση και αποθήκευση του αριστερού μισού στην μεταβλητή lefthalf και του δεξιού μισού στην righthalf.

RoundShift(key): Αριστερή κυκλική ολίσθηση κατά ένα bit.

XOR(list1,list2): Λογική πράξη XOR μεταξύ bits είτε ακολουθιών bits.

BintoDec(binaryNum): Μετατροπή από δυαδικά ψηφία σε δεκαδικά.

SboxMatrixChoise(fourBitKey,Sbox): Επιλογή των κατάλληλων δύο bit(ένα κελί) από το S0 ή S1 box και επιστροφή της τιμής του.

correct_variable(value): Αποθήκευση της σωστής τιμής των μεταβλητών που αλλάζουν συνεχόμενα λόγω global.

keyProduction(): Διαδικασία παραγωγής κλειδιών K1 και K2

list_group_8(list): Ομαδοποίηση των bits σε ομάδες των 8-bits και αποθήκευση σε λίστα.

Encryption_SDES(key,initPermBits): Διαδικασία κρυπτογράφησης και αποκρυπτογράφησης ανάλογα με την είσοδο και την φορά επανάληψης της συνάρτησης αυτής.

main(): Βασική κινητήρια συνάρτηση για την υλοποίηση του S-DES που καθορίζει το πόσες φορές θα εκτελεστεί η **Encryption_SDES(key,initPermBits)** , το που θα αποθηκευτούν τα αποτελέσματα του SDES και το πώς θα αποθηκευτούν ενδεικτικά τα αποτελέσματα των 8 πρώτων bits για να εμφανιστούν σαν παράδειγμα στο ui.

Αλγόριθμος S-DES:

Simplified Data Encryption Standard (S-DES) είναι ένας αλγόριθμος κρυπτογράφησης μπλοκ. Παίρνει σαν είσοδο ένα μπλοκ 8-bit απλού κειμένου ή 8-bit και ένα κλειδί μεγέθους 10-bit και τελικά παράγει ένα κρυπτογραφημένο μπλοκ 8-bit cipher-text ως έξοδο. Πρόκειται για ένα συμμετρικό κρυπτογράφο που έχει δύο επαναλήψεις για να παραχθεί το τελικό κρυπτογραφημένο 8 bit cipher μία με το κλειδί K1 και μία με το

K2 ενώ και για την αποκρυπτογράφηση του χρειάζονται επίσης άλλες δυο επανάληψης με πρώτα το κλειδί K2 και μετά το K1. Για την κρυπτογράφηση και την αποκρυπτογράφηση χρησιμοποιούνται συγκεκριμένες μεταθέσεις των bits που ορίζονται από τον ίδιο τον αλγόριθμο. Για τις μεταθέσεις αυτές υπάρχουν συγκεκριμένοι πίνακες που ορίζουν την σειρά με την οποία θα γίνονται: P10, P8, P4 EP(Expand&Permutate),IP ,IP^{-1} και οι πίνακες S0-box και S1-box.

P10:

0	1	2	3	4	5	6	7	8	9
1	4	1	6	3	9	0	8	7	5

P8:

0	1	2	3	4	5	6	7
5	2	6	3	7	4	9	8

P4:

0	1	2	3
1	3	2	0

IP

0	1	2	3	4	5	6	7
1	5	2	0	3	7	4	6

IP^{-1}

0	1	2	3	4	5	6	7
3	0	2	4	6	1	7	5

EP

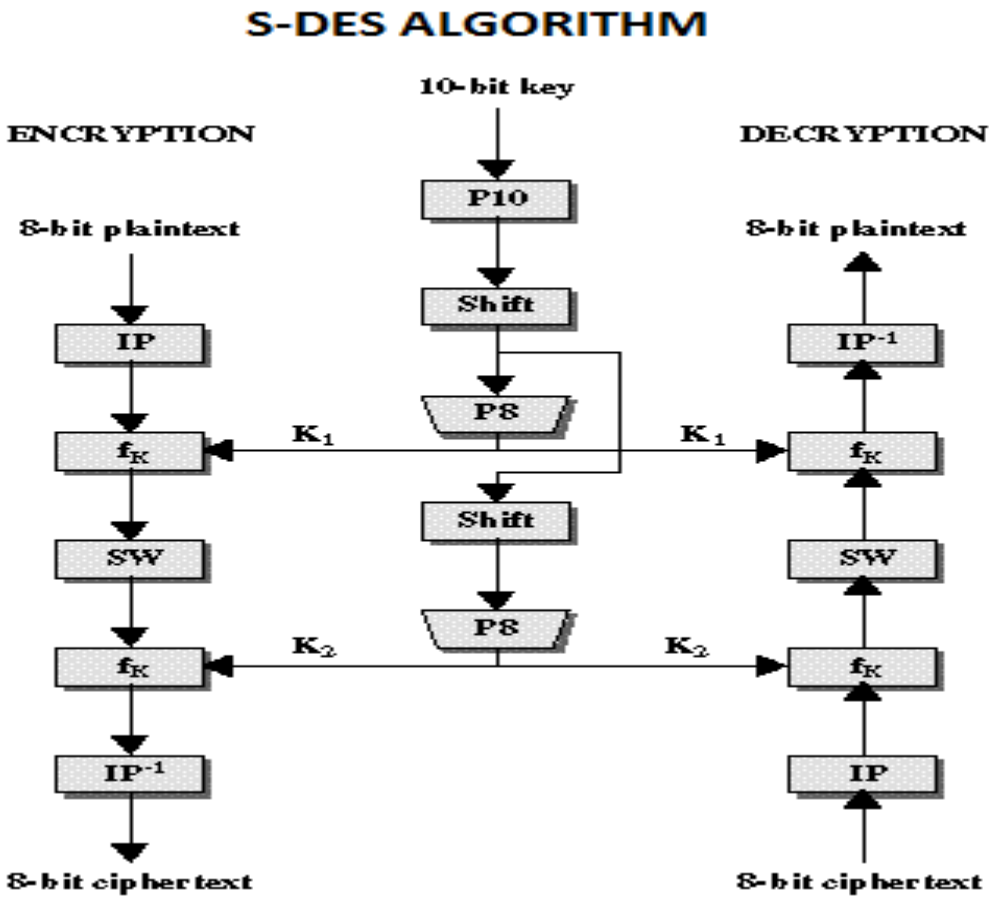
0		1		2		3	
3	0	1	2	1	2	3	0

S-0

Col	0	1	2	3
Rows				
0	01	00	11	10
1	11	10	01	00
2	00	10	01	11
3	11	01	11	10

S-1

Col	0	1	2	3
Rows				
0	00	01	10	11
1	10	00	01	11
2	11	00	01	00
3	10	01	00	11

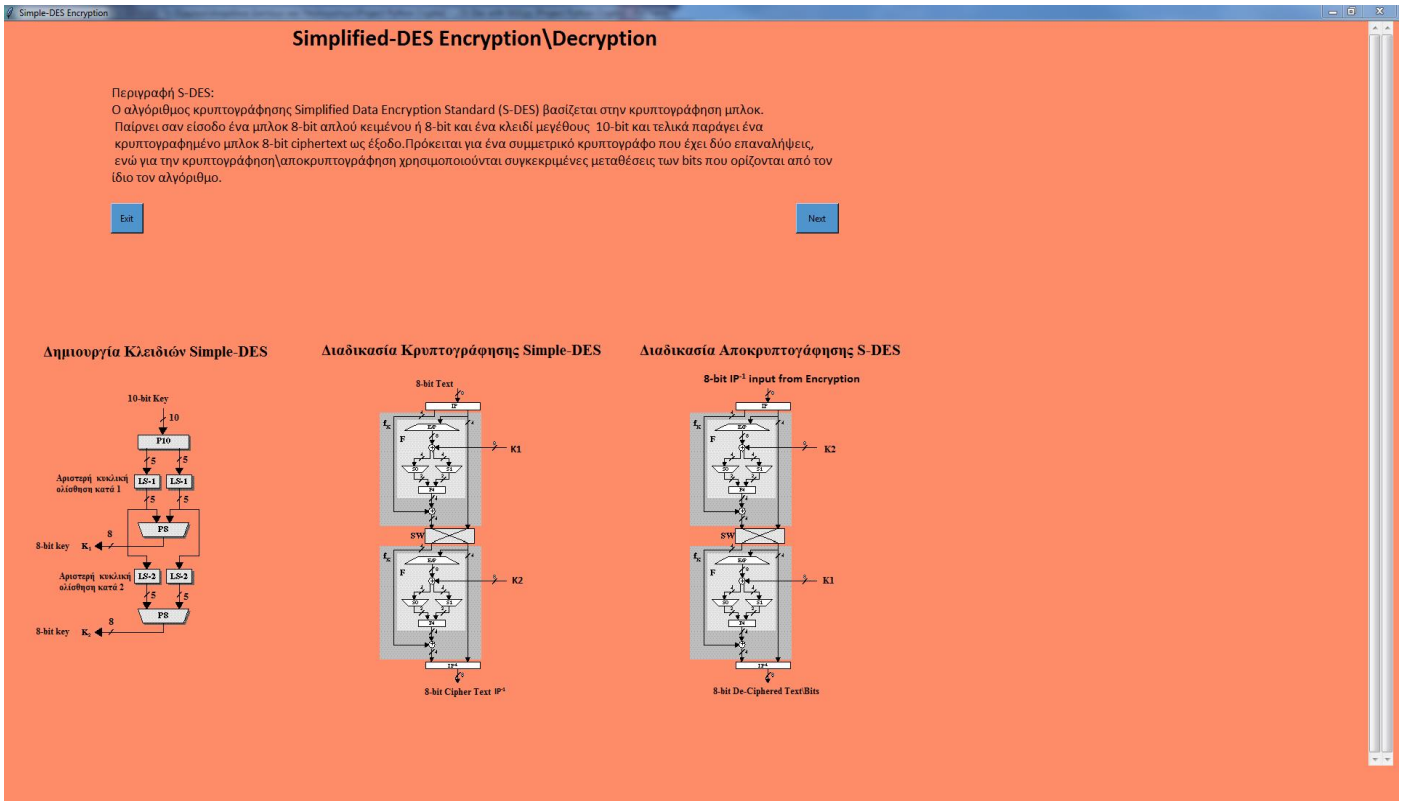


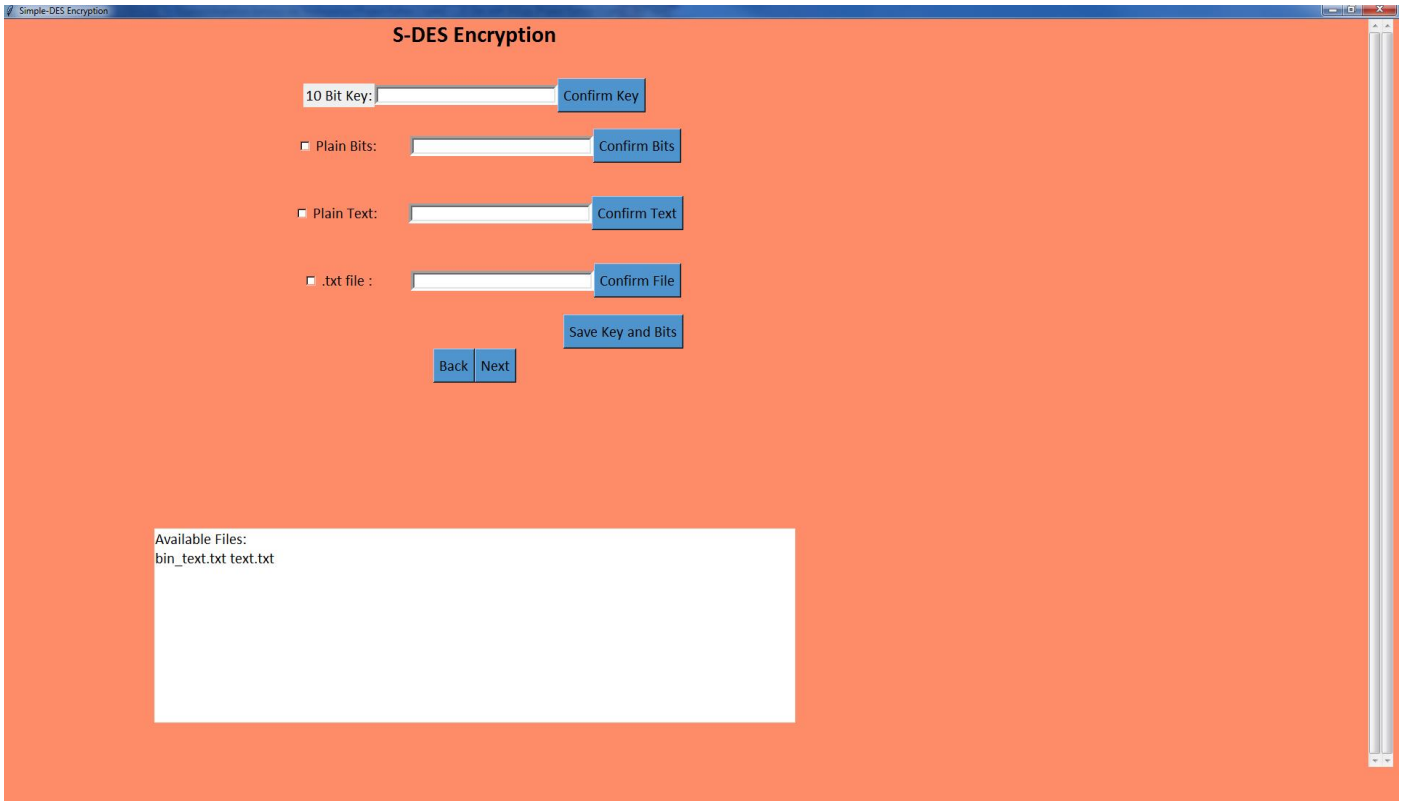
Γραφικό Περιβάλλον GUI:

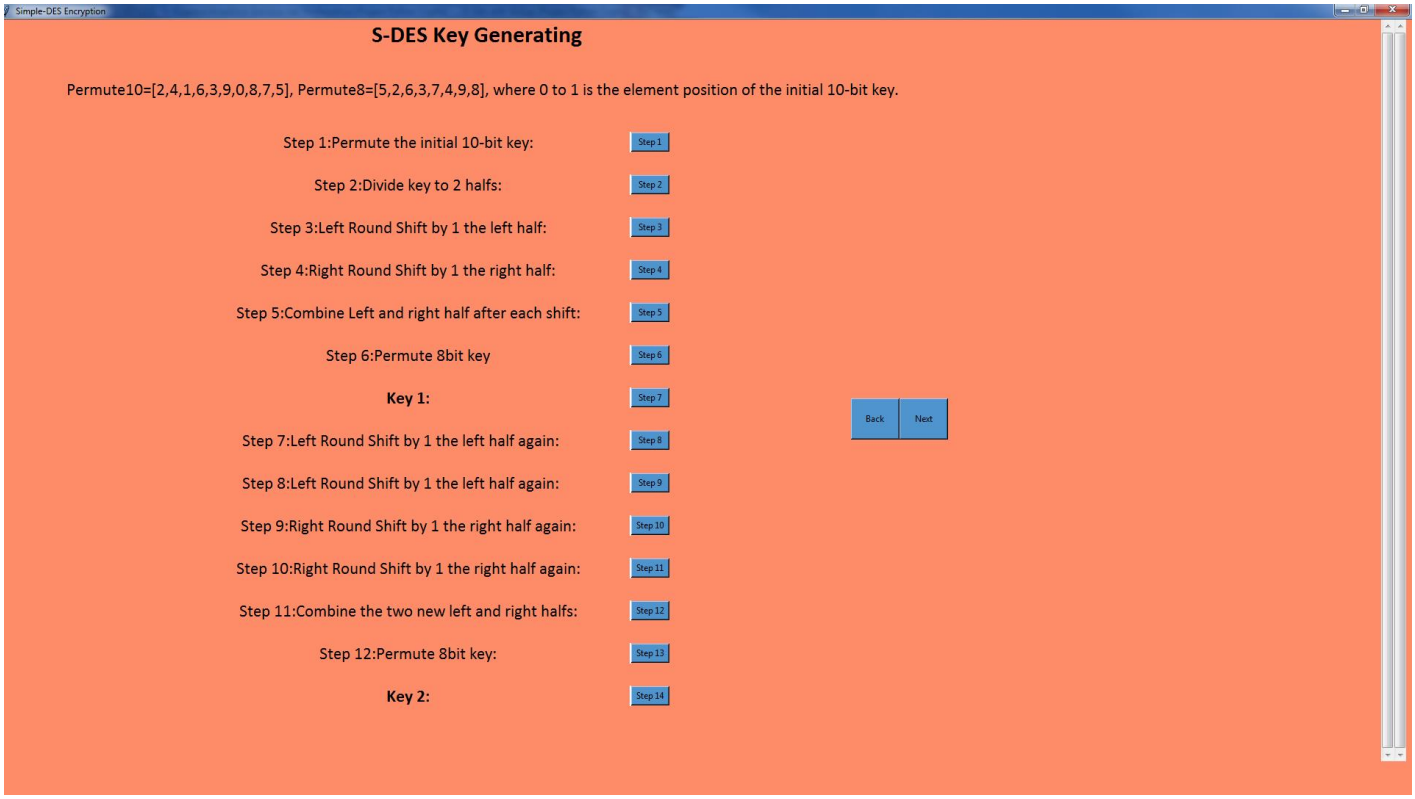
Το γραφικό μας περιβάλλον υλοποιήθηκε με την βοήθεια του tkinter. Έχουμε δημιουργήσει ένα παράθυρο πάνω στο οποίο αλλάζουν τα frames με βάση τα κουμπιά Back Next ενώ κλείνει με το κουμπί Exit. Αυτό το window έχει ένα βασικό frame στο οποίο ανήκουν όλα τα υπόλοιπα frames βρίσκεται μέσα σε ένα canvas για να μπορέσουμε να δώσουμε την δυνατότητα να έχει scrollbars για τον χ και y άξονα στη περίπτωση που η ανάλυση στο υπολογιστή του χρήστη είναι μικρότερη από 1920x1080. Το γραφικό περιβάλλον συνολικά αποτελείτε από 8 διαφάνειες οι οποίες αποτελούνται από labels ,texts ,checkboxes και buttons όλα αυτά για να εξασφαλιστεί η ευχάριστη και εύκολη λειτουργεία από τον χρήστη. Παρακάτω παρατίθενται φωτογραφίες του προγράμματος εν ώρα λειτουργίας σε υπολογιστή με ανάλυση 1920x1080

Screenshots Προγράμματος:

Αρχική εικόνα των σελίδων:






Simple-DES Encryption
S-DES Key Generating
Permute10=[2,4,1,6,3,9,0,8,7,5], Permute8=[5,2,6,3,7,4,9,8], where 0 to 1 is the element position of the initial 10-bit key.
Step 1:Permute the initial 10-bit key:
Step 1
Step 2:Divide key to 2 halfs:
Step 2
Step 3:Left Round Shift by 1 the left half:
Step 3
Step 4:Right Round Shift by 1 the right half:
Step 4
Step 5:Combine Left and right half after each shift:
Step 5
Step 6:Permute 8bit key
Step 6
Key 1:
Step 7
Back
Next
Step 7:Left Round Shift by 1 the left half again:
Step 8
Step 8:Left Round Shift by 1 the left half again:
Step 9
Step 9:Right Round Shift by 1 the right half again:
Step 10
Step 10:Right Round Shift by 1 the right half again:
Step 11
Step 11:Combine the two new left and right halfs:
Step 12
Step 12:Permute 8bit key:
Step 13
Key 2:
Step 14


Simple-DES Encryption
S-DES Encryption
Permute8=[5,2,6,3,7,4,9,8], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Loop 1 for Key 1
Step 1:Initial 8-bit input:
Step 1
Step 2:Permute the initial 8-bit input:
Step 2
Step 3:Divide to 2 halfs:
Step 3
Step 4:Expand And Permutate right half:
Step 4
Step 5:Previous output XOR key 1:
Step 5
Step 6:Divide to 2 halfs:
Step 6
Step 7:Left S0-box:
Step 7
Back
Next
Step 8:Right S1-box:
Step 8
Step 9:Combine the two new left and right 2-bits:
Step 9
Step 10:Permute 4 last output:
Step 10
Step 11:Last Output XOR Left half of initial input:
Step 11
Step 12:Combine last output right half of initial input:
Step 12
Step 13:Divide to 2 halfs:
Step 13
Step 14:Swap Left & Right half's Positions:
Step 14


Simple-DES Encryption
S-DES Encryption
(IP^-1)IP8-inversed=[3,0,2,4,6,1,7,5], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Loop 2 for key 2
Step 1:8-bit input after SW(swap):
Step 1
Step 2:Divide to 2 halfs:
Step 2
Step 3::Expand And Permutate right half:
Step 3
Step 4:Previous output XOR key 2:
Step 4
Step 5:Divide to 2 halfs:
Step 5
Step 6:Left S0-box:
Step 6
Step 7:Right S1-box:
Step 7
Step 8:Combine the two new left and right 2-bits:
Step 8
Step 9:Permute 4 last output:
Step 9
Step 10::Last Output XOR Left half of initial input:
Step 10
Step 11:Combine last output right half of initial input:
Step 11
Step 12:PermuteIP^-1 8-bit-> Cipher text:
Step 12
Back
Next

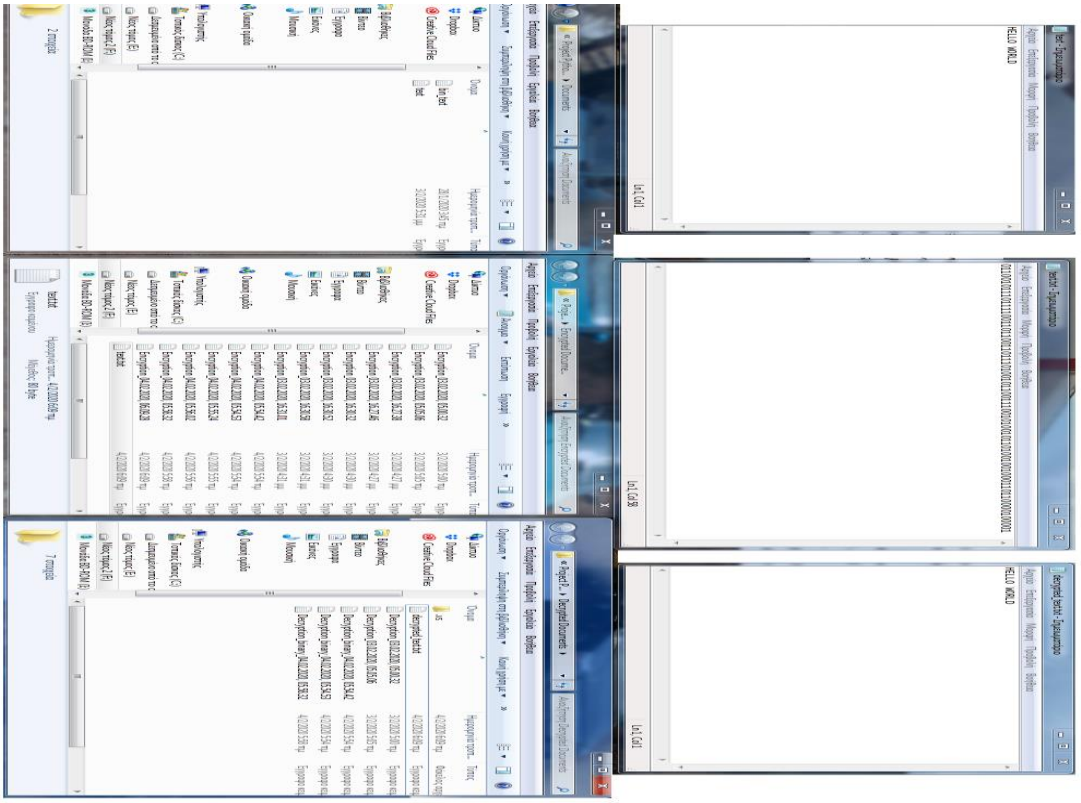

Simple-DES Encryption
S-DES Decryption
Permute8=[5,2,6,3,7,4,9,8], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Step 1:Cipher 8-bit input((IP^-1) of encryption exit):
Step 1
Step 2:Permute the Cipher 8-bit input:
Step 2
Step 3:Divide to 2 halfs:
Step 3
Step 4:Expand And Permutate right half:
Step 4
Step 5:Previous output XOR key 2:
Step 5
Step 6:Divide to 2 halfs:
Step 6
Step 7:Left S0-box:
Step 7
Step 8:Right S1-box:
Step 8
Step 9:Combine the two new left and right 2-bits:
Step 9
Step 10:Permute 4 last output:
Step 10
Step 11:Last Output XOR Left half of initial cipher input:
Step 11
Step 12:Combine last output right half of initial cipher input:
Step 12
Step 13:Divide to 2 halfs:
Step 13
Step 14:Swap Left & Right half's Positions:
Step 14
Back
Next

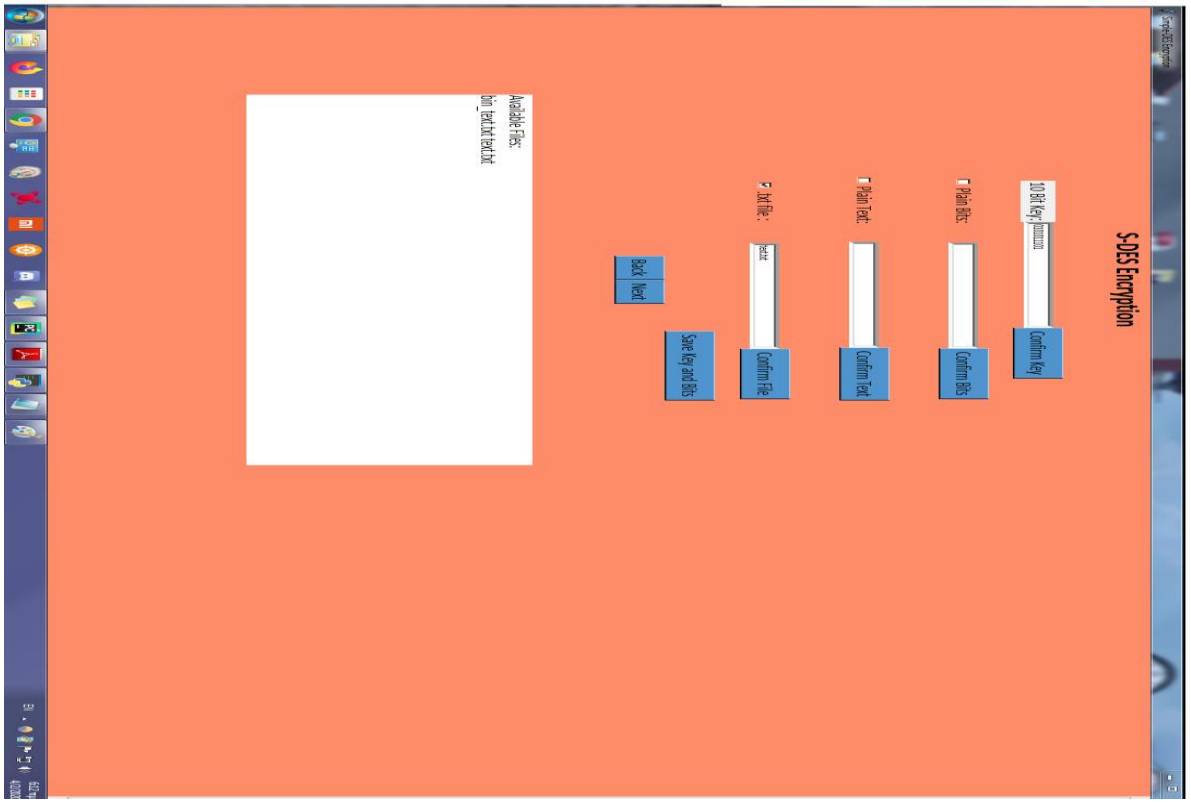

Simple-DES Encryption
S-DES Decryption
(IP^-1)IP8-inversed=[3,0,2,4,6,1,7,5], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Loop 2 for key 1
Step 1:8-bit input after SW(swap):
Step 1
Step 2:Divide to 2 halfs:
Step 2
Step 3::Expand And Permutate right half:
Step 3
Step 4:Previous output XOR key 2:
Step 4
Step 5:Divide to 2 halfs:
Step 5
Step 6:Left S0-box:
Step 6
Step 7:Right S1-box:
Step 7
Step 8:Combine the two new left and right 2-bits:
Step 8
Step 9:Permute 4 last output:
Step 9
Step 10::Last Output XOR Left half of initial input:
Step 10
Step 11:Combine last output right half of initial input:
Step 11
Step 12: PermuteIP^-1 8-bit-> -> De-Ciphered text/bits:
Step 12
Back
Next

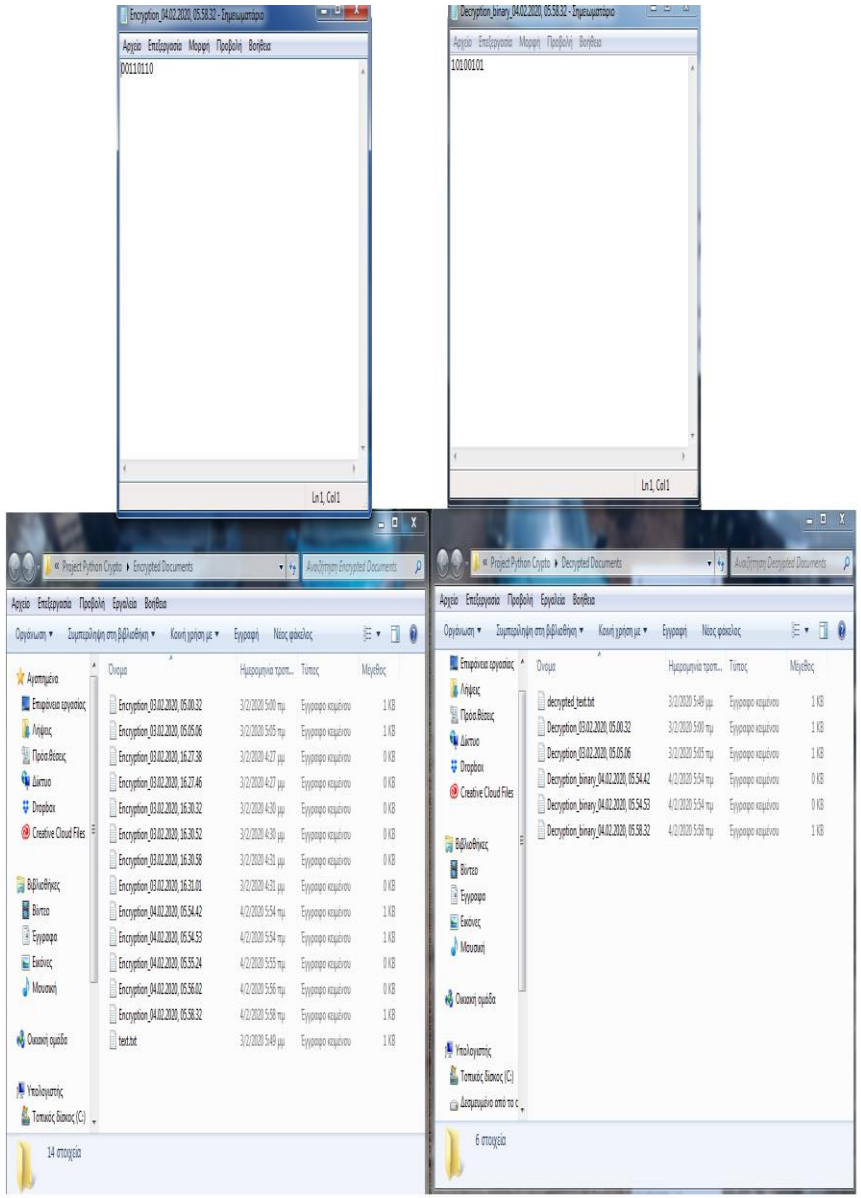

Simple-DES Encryption
S-DES Encryption/Decryption Finished
Please choose one of the following options Back or Exit to continue or exit the program!!!
Note: You can check the Documents, Encrypted Documents and Decrypted Documents folders for the results of S-DES, at the directory:
F:\CEID\CEID 7ο Εξαμηνο\Ασφάλεια Δικτύων και Υπολογιστών\Project Python Crypto
Back
Exit
ΒΑΖΑΙΟΣ ΣΤΥΛΙΑΝΟΣ Α.Μ.:1054284 CEID

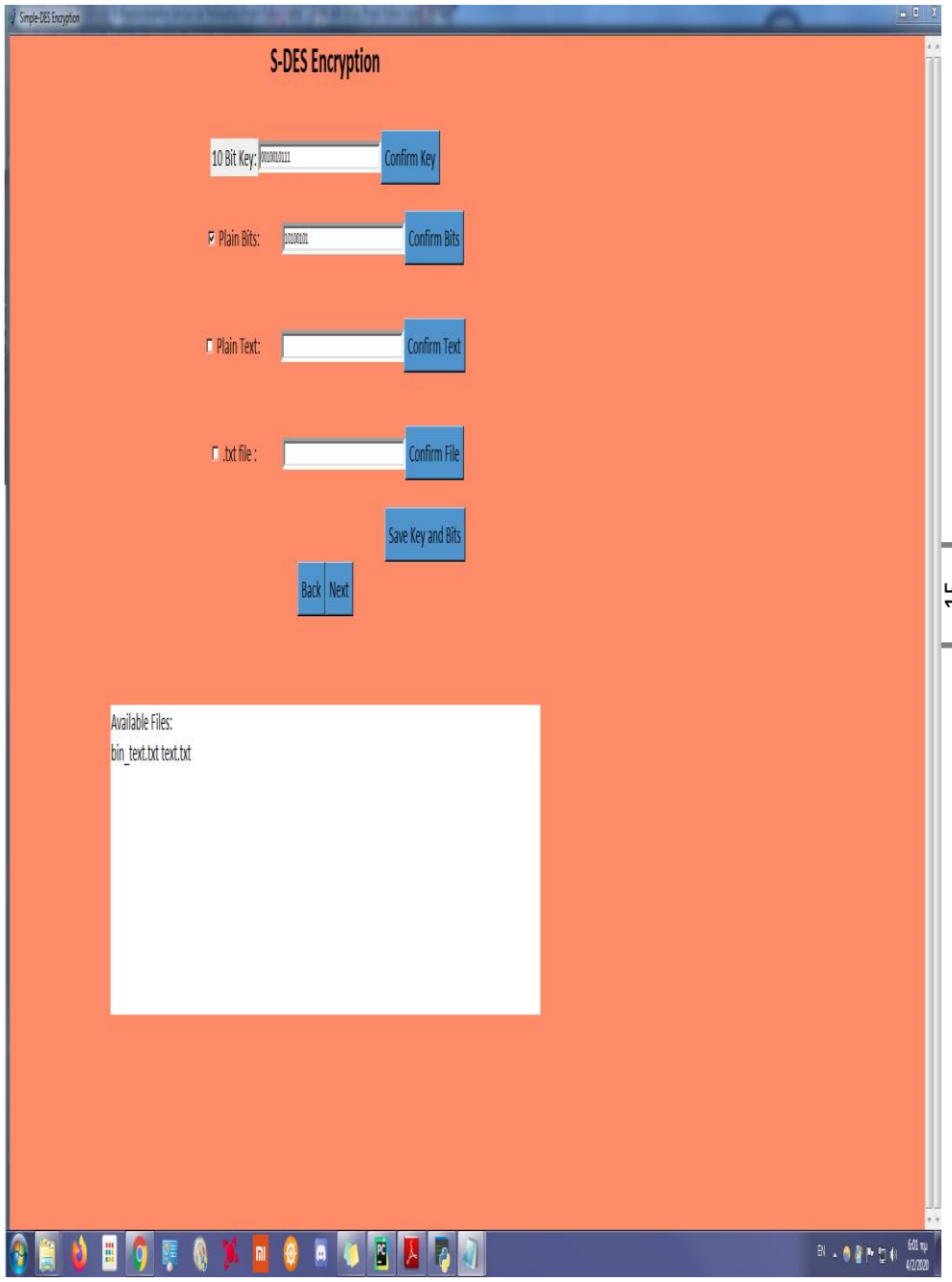
Αποτελέσματα με τυχαία είσοδο:

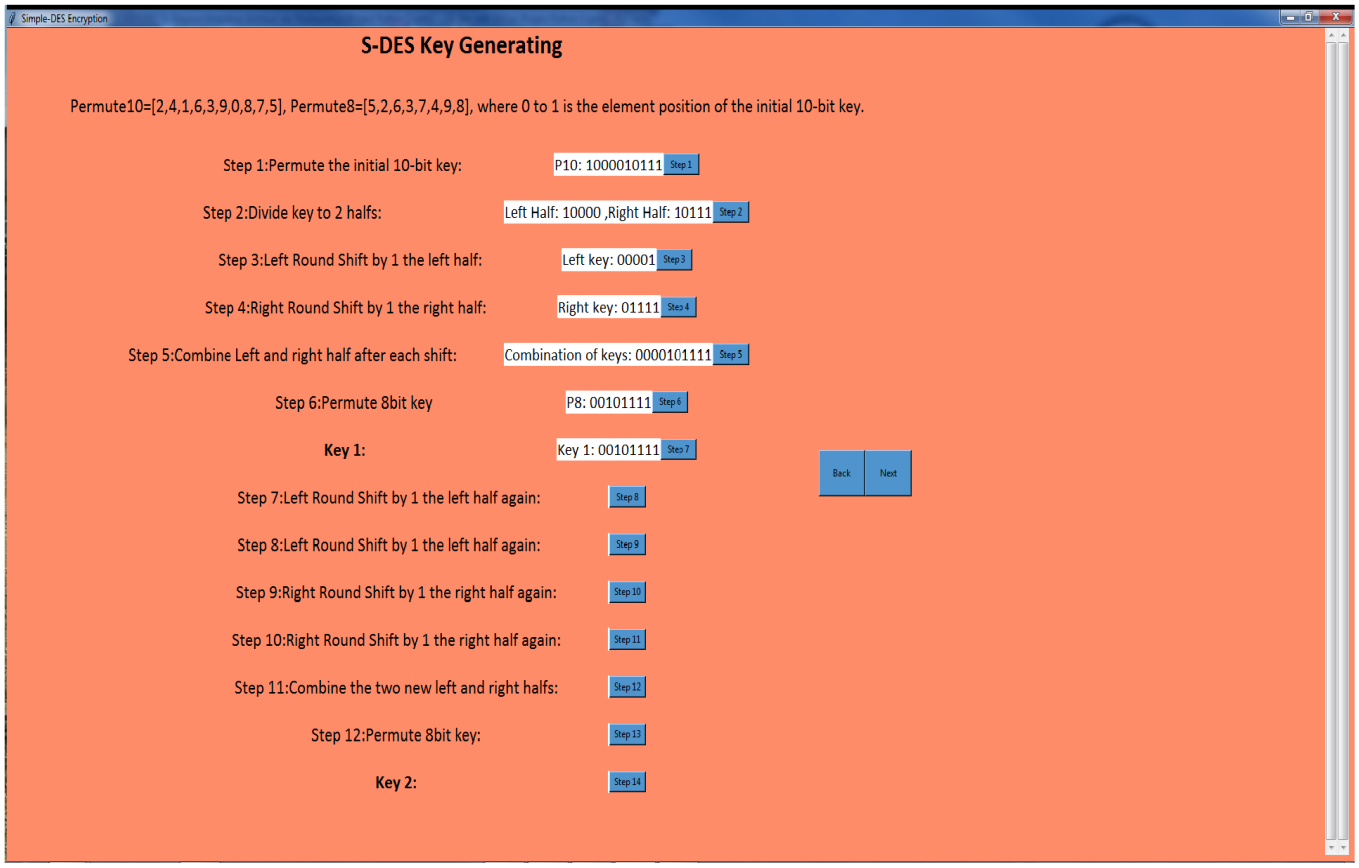
Αρχικά η είσοδος είναι από το αρχείο text.txt
Και έπειτα βάζουμε μια τυχαία ακολουθία 8 bit 10100101 με κλειδί 10bit 0010010111.
Με βάση αυτές τις εισόδους βλέπουμε και τα αντίστοιχα αρχεία που προκύπτουν.

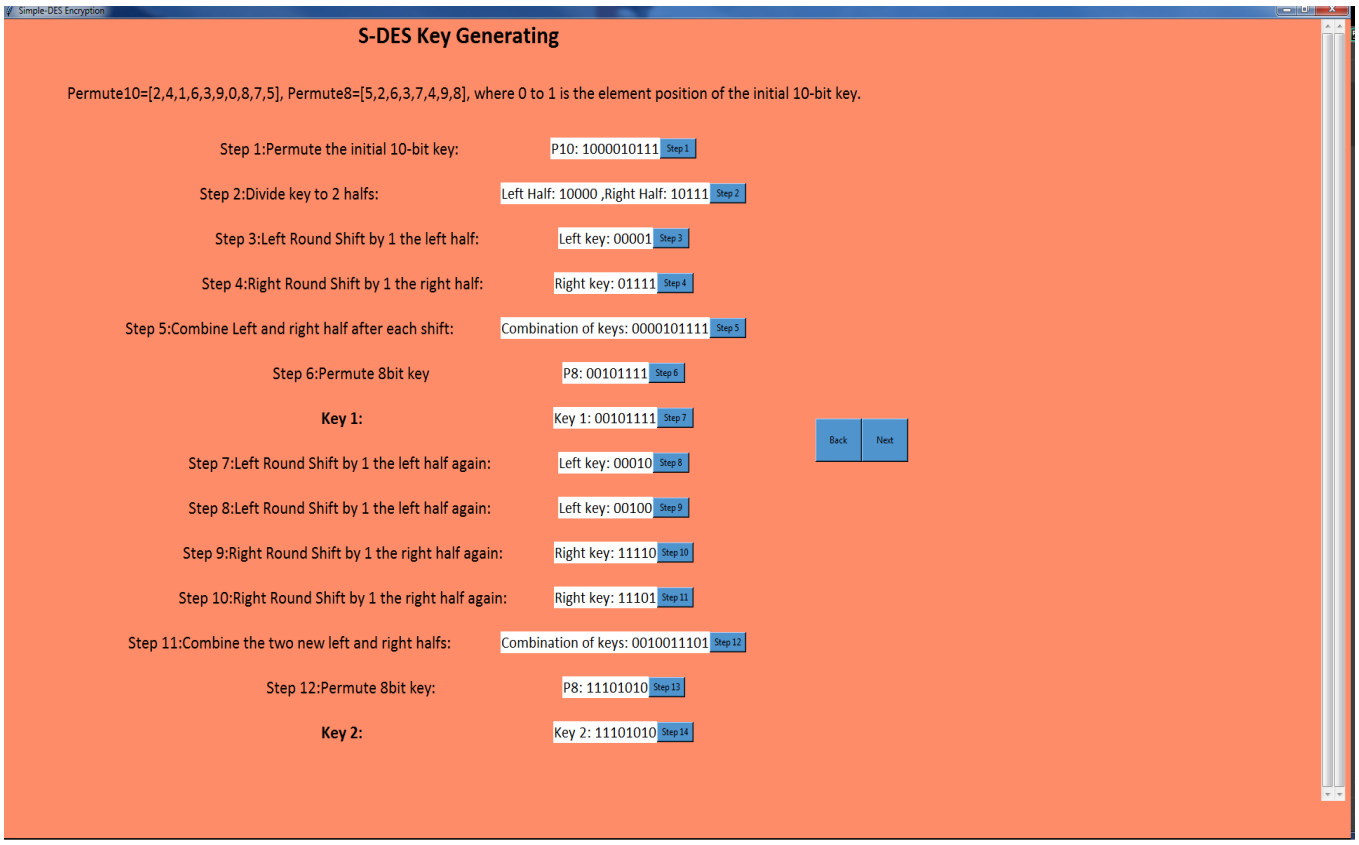


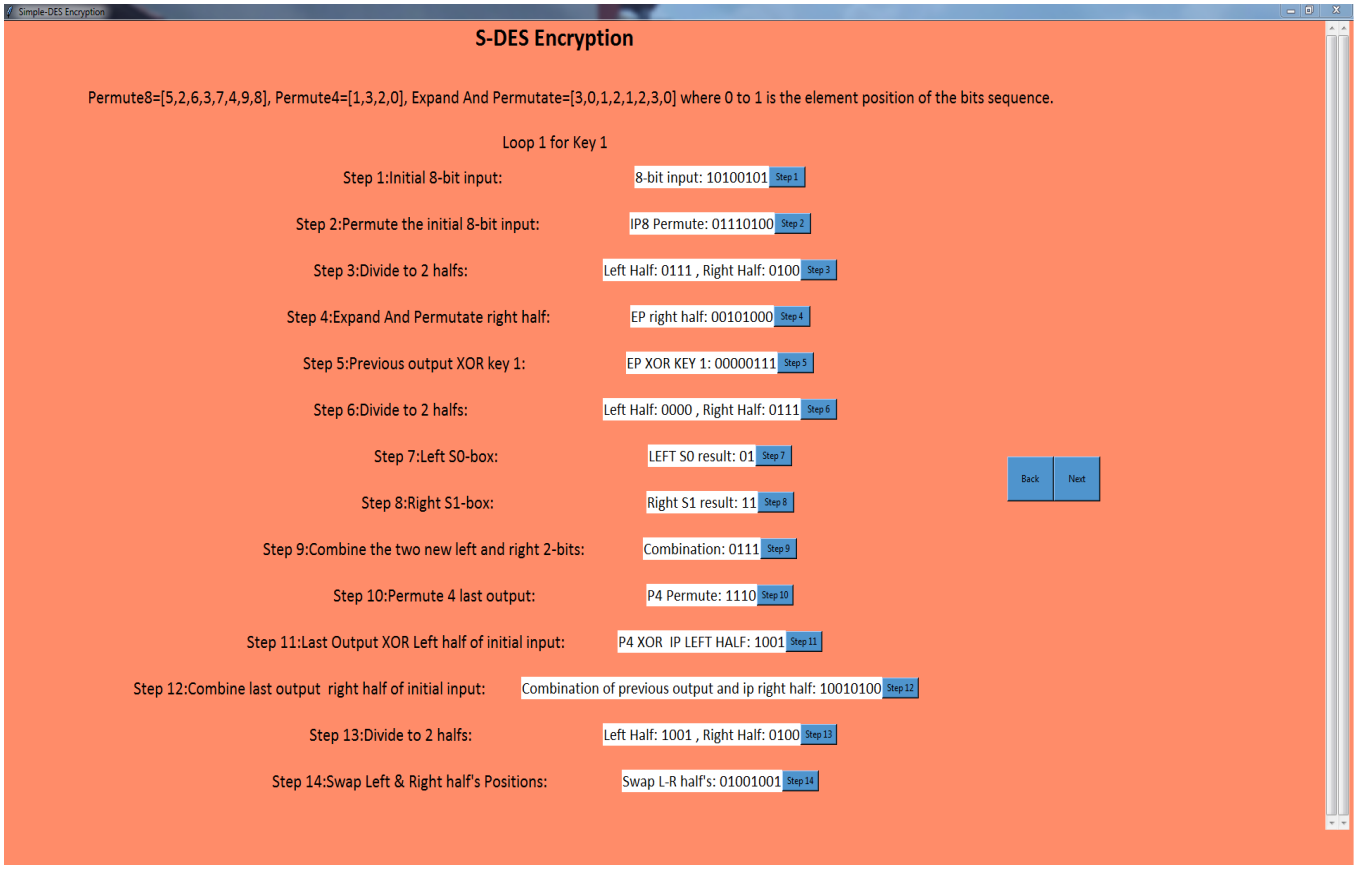


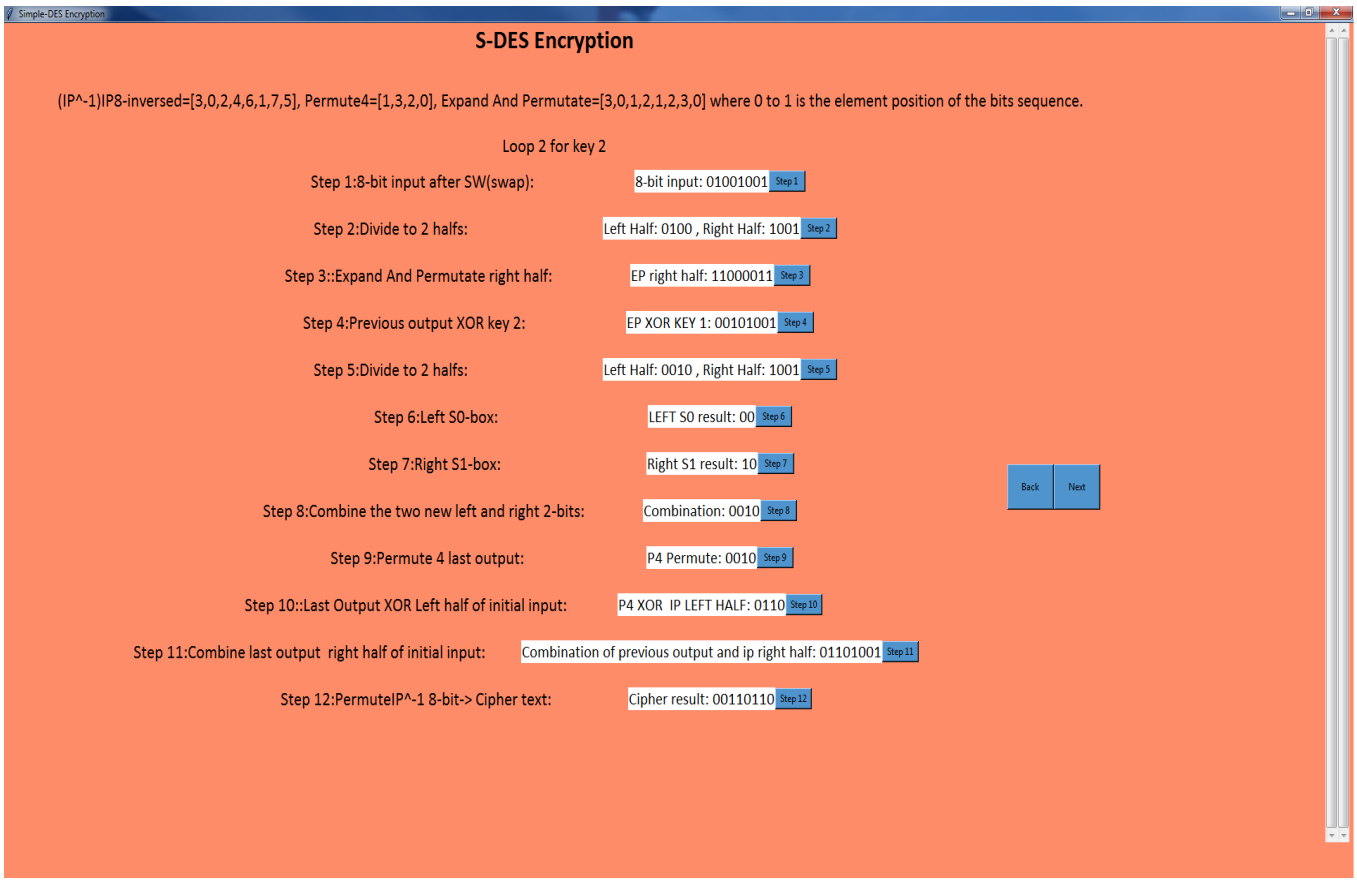

Encryption_04.02.2020_05.58.32 - Σημειωματάριο
Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια
00110110
Ln 1, Col 1
Decryption_binary_04.02.2020_05.58.32 - Σημειωματάριο
Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια
10100101
Ln 1, Col 1
Project Python Crypto ▸ Encrypted Documents
Project Python Crypto ▸ Decrypted Documents
14 στοιχεία
6 στοιχεία

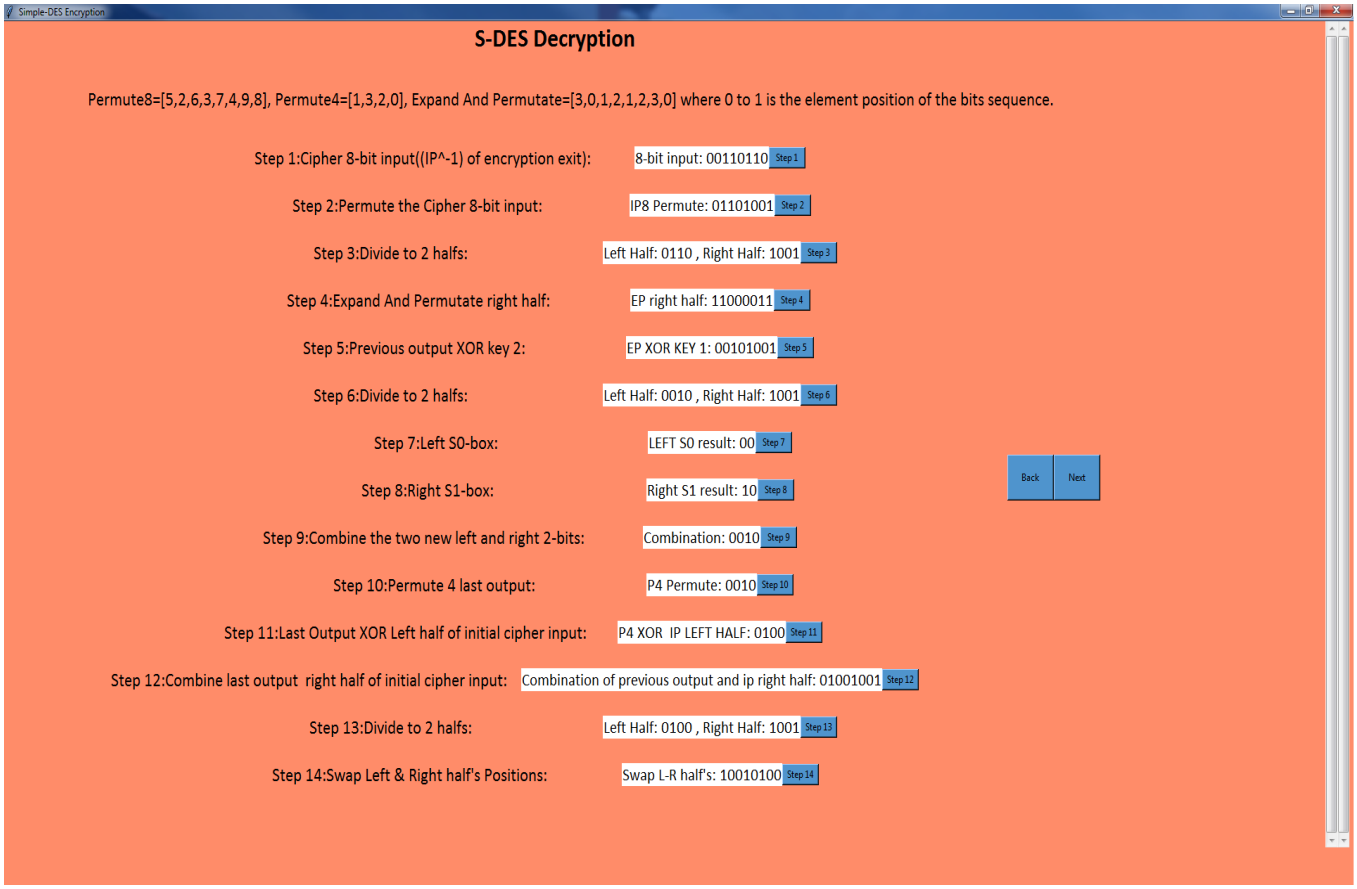

Simple-DES Encryption
S-DES Encryption
10 Bit Key:
Confirm Key
Plain Bits:
Confirm Bits
Plain Text:
Confirm Text
.txt file :
Confirm File
Save Key and Bits
Back
Next
Available Files:
bin_text.txt text.txt

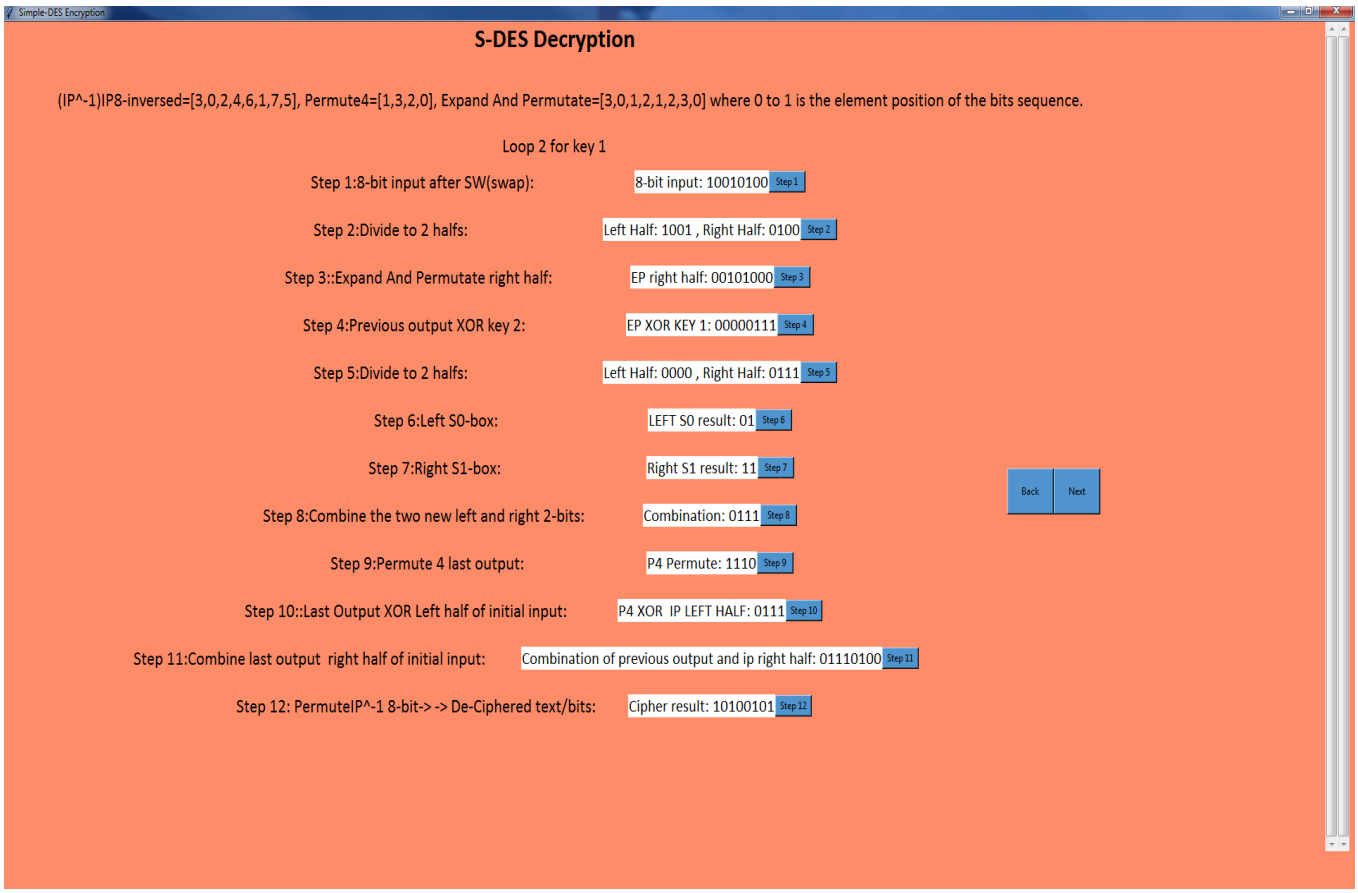

Simple-DES Encryption
S-DES Key Generating
Permute10=[2,4,1,6,3,9,0,8,7,5], Permute8=[5,2,6,3,7,4,9,8], where 0 to 1 is the element position of the initial 10-bit key.
Step 1:Permute the initial 10-bit key:
P10: 1000010111
Step 1
Step 2:Divide key to 2 halfs:
Left Half: 10000 ,Right Half: 10111
Step 2
Step 3:Left Round Shift by 1 the left half:
Left key: 00001
Step 3
Step 4:Right Round Shift by 1 the right half:
Right key: 01111
Step 4
Step 5:Combine Left and right half after each shift:
Combination of keys: 0000101111
Step 5
Step 6:Permute 8bit key
P8: 00101111
Step 6
Key 1:
Key 1: 00101111
Step 7
Back
Next
Step 7:Left Round Shift by 1 the left half again:
Step 8
Step 8:Left Round Shift by 1 the left half again:
Step 9
Step 9:Right Round Shift by 1 the right half again:
Step 10
Step 10:Right Round Shift by 1 the right half again:
Step 11
Step 11:Combine the two new left and right halfs:
Step 12
Step 12:Permute 8bit key:
Step 13
Key 2:
Step 14


Simple-DES Encryption
S-DES Key Generating
Permute10=[2,4,1,6,3,9,0,8,7,5], Permute8=[5,2,6,3,7,4,9,8], where 0 to 1 is the element position of the initial 10-bit key.
Step 1:Permute the initial 10-bit key:
P10: 1000010111
Step 1
Step 2:Divide key to 2 halfs:
Left Half: 10000 ,Right Half: 10111
Step 2
Step 3:Left Round Shift by 1 the left half:
Left key: 00001
Step 3
Step 4:Right Round Shift by 1 the right half:
Right key: 01111
Step 4
Step 5:Combine Left and right half after each shift:
Combination of keys: 0000101111
Step 5
Step 6:Permute 8bit key
P8: 00101111
Step 6
Key 1:
Key 1: 00101111
Step 7
Back
Next
Step 7:Left Round Shift by 1 the left half again:
Left key: 00010
Step 8
Step 8:Left Round Shift by 1 the left half again:
Left key: 00100
Step 9
Step 9:Right Round Shift by 1 the right half again:
Right key: 11110
Step 10
Step 10:Right Round Shift by 1 the right half again:
Right key: 11101
Step 11
Step 11:Combine the two new left and right halfs:
Combination of keys: 0010011101
Step 12
Step 12:Permute 8bit key:
P8: 11101010
Step 13
Key 2:
Key 2: 11101010
Step 14

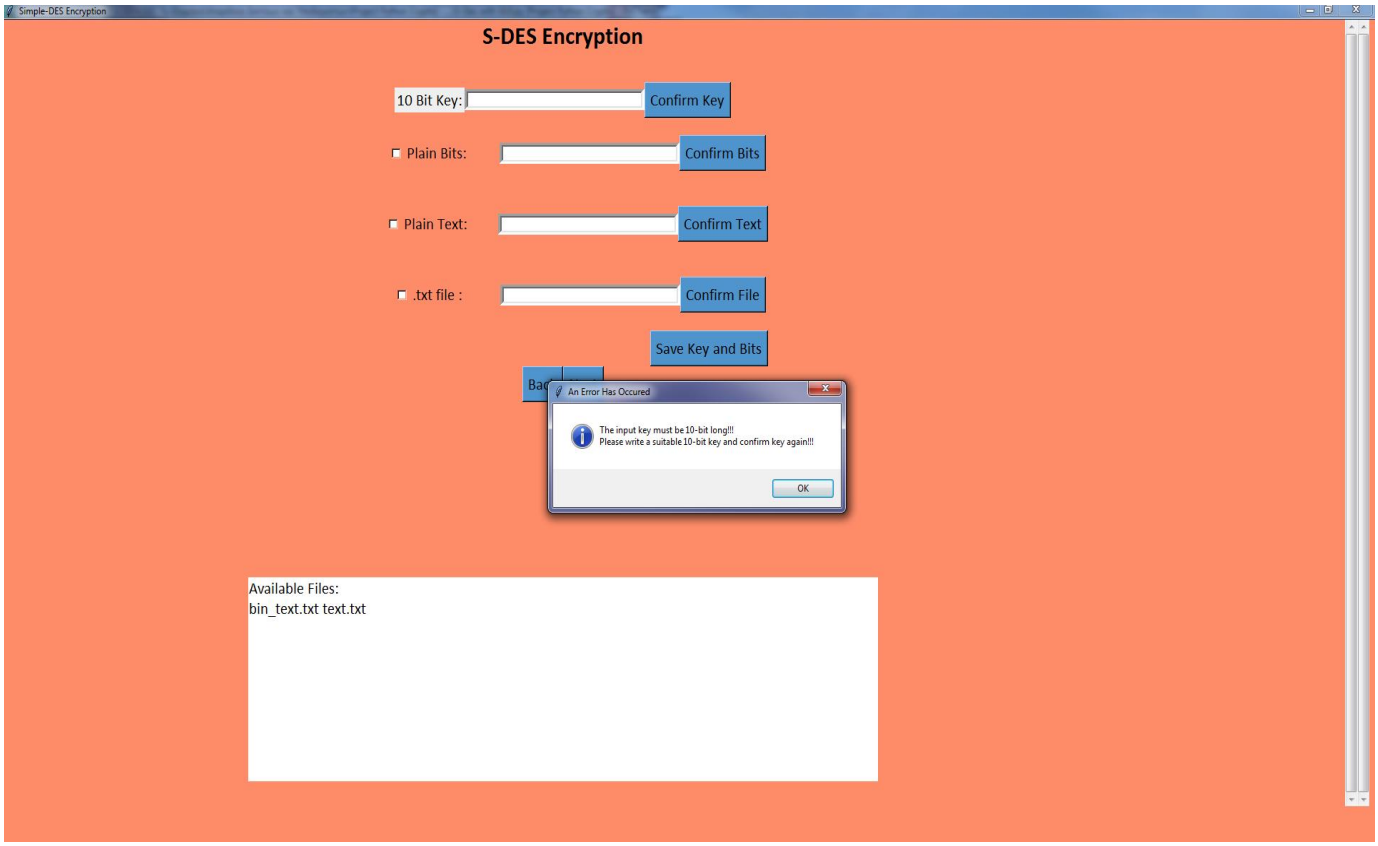

Simple-DES Encryption
S-DES Encryption
Permute8=[5,2,6,3,7,4,9,8], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Loop 1 for Key 1
Step 1:Initial 8-bit input:
8-bit input: 10100101
Step 1
Step 2:Permute the initial 8-bit input:
IP8 Permute: 01110100
Step 2
Step 3:Divide to 2 halfs:
Left Half: 0111 , Right Half: 0100
Step 3
Step 4:Expand And Permutate right half:
EP right half: 00101000
Step 4
Step 5:Previous output XOR key 1:
EP XOR KEY 1: 00000111
Step 5
Step 6:Divide to 2 halfs:
Left Half: 0000 , Right Half: 0111
Step 6
Step 7:Left S0-box:
LEFT S0 result: 01
Step 7
Step 8:Right S1-box:
Right S1 result: 11
Step 8
Step 9:Combine the two new left and right 2-bits:
Combination: 0111
Step 9
Step 10:Permute 4 last output:
P4 Permute: 1110
Step 10
Step 11:Last Output XOR Left half of initial input:
P4 XOR IP LEFT HALF: 1001
Step 11
Step 12:Combine last output right half of initial input:
Combination of previous output and ip right half: 10010100
Step 12
Step 13:Divide to 2 halfs:
Left Half: 1001 , Right Half: 0100
Step 13
Step 14:Swap Left & Right half's Positions:
Swap L-R half's: 01001001
Step 14
Back
Next

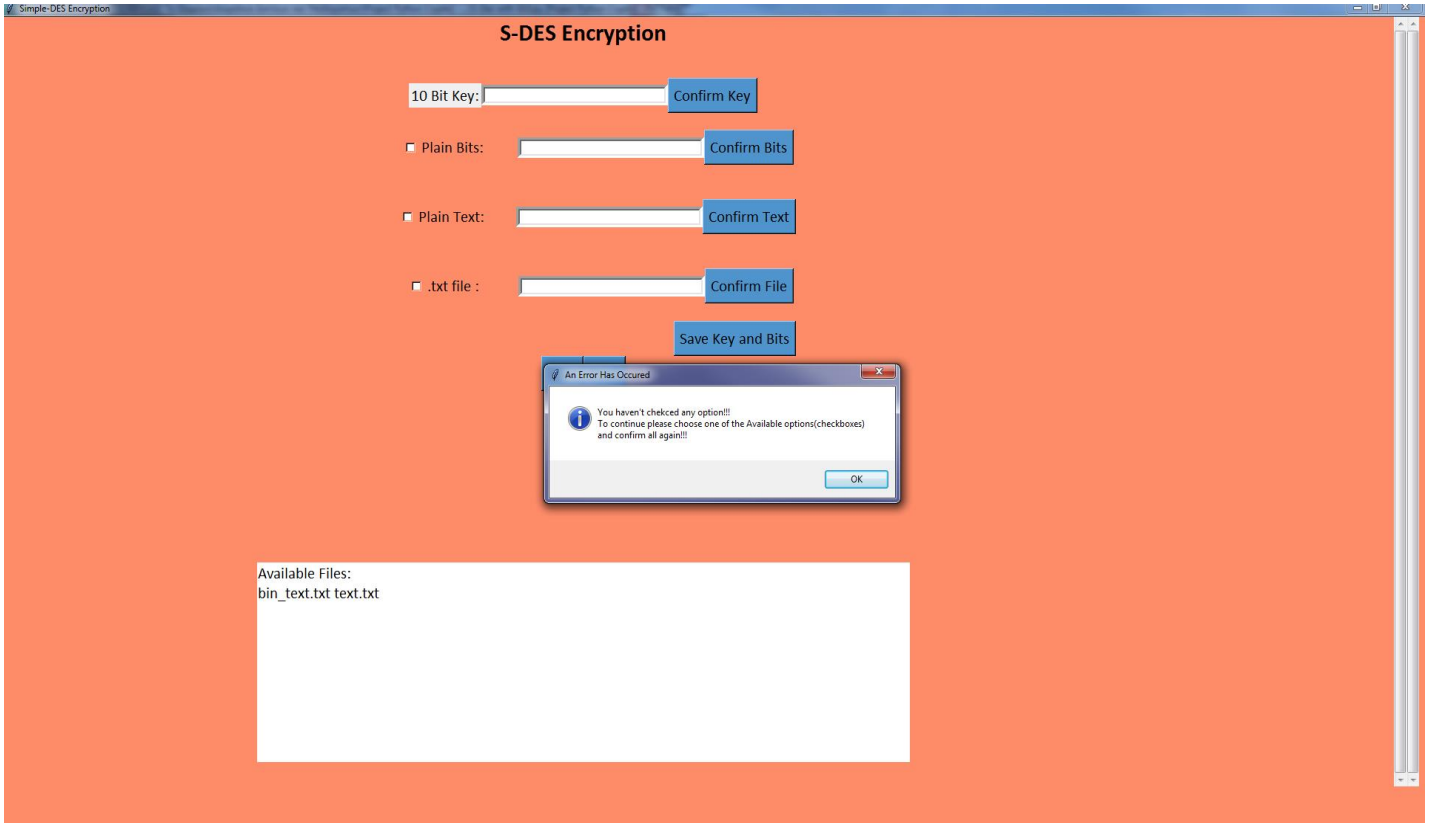

Simple-DES Encryption
S-DES Encryption
(IP^-1)IP8-inversed=[3,0,2,4,6,1,7,5], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Loop 2 for key 2
Step 1:8-bit input after SW(swap):
8-bit input: 01001001
Step 1
Step 2:Divide to 2 halfs:
Left Half: 0100 , Right Half: 1001
Step 2
Step 3::Expand And Permutate right half:
EP right half: 11000011
Step 3
Step 4:Previous output XOR key 2:
EP XOR KEY 1: 00101001
Step 4
Step 5:Divide to 2 halfs:
Left Half: 0010 , Right Half: 1001
Step 5
Step 6:Left S0-box:
LEFT S0 result: 00
Step 6
Step 7:Right S1-box:
Right S1 result: 10
Step 7
Step 8:Combine the two new left and right 2-bits:
Combination: 0010
Step 8
Step 9:Permute 4 last output:
P4 Permute: 0010
Step 9
Step 10::Last Output XOR Left half of initial input:
P4 XOR IP LEFT HALF: 0110
Step 10
Step 11:Combine last output right half of initial input:
Combination of previous output and ip right half: 01101001
Step 11
Step 12:PermuteIP^-1 8-bit-> Cipher text:
Cipher result: 00110110
Step 12
Back
Next

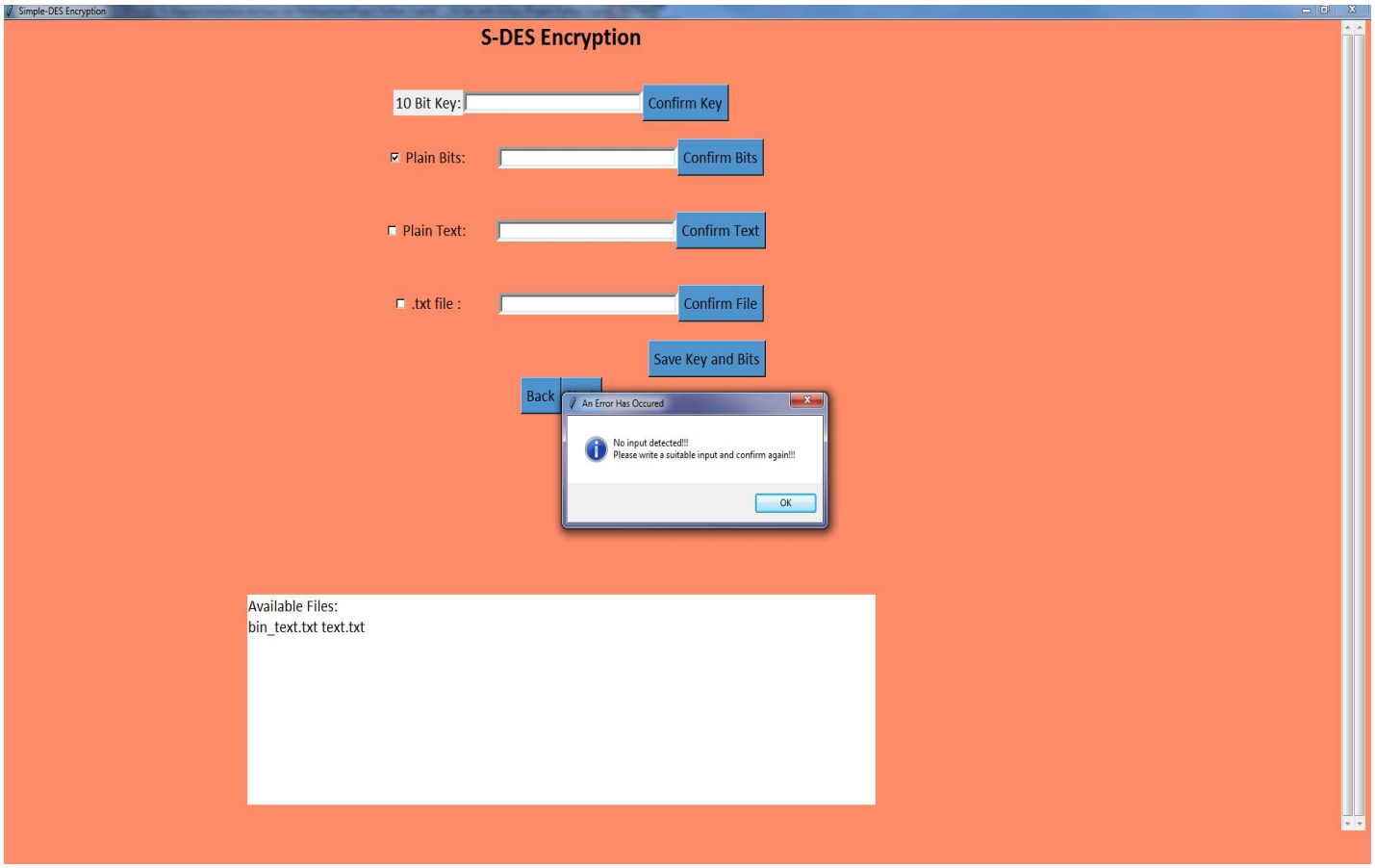

Simple-DES Encryption
S-DES Decryption
Permute8=[5,2,6,3,7,4,9,8], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Step 1:Cipher 8-bit input((IP^-1) of encryption exit):
8-bit input: 00110110
Step 1
Step 2:Permute the Cipher 8-bit input:
IP8 Permute: 01101001
Step 2
Step 3:Divide to 2 halfs:
Left Half: 0110 , Right Half: 1001
Step 3
Step 4:Expand And Permutate right half:
EP right half: 11000011
Step 4
Step 5:Previous output XOR key 2:
EP XOR KEY 1: 00101001
Step 5
Step 6:Divide to 2 halfs:
Left Half: 0010 , Right Half: 1001
Step 6
Step 7:Left S0-box:
LEFT S0 result: 00
Step 7
Step 8:Right S1-box:
Right S1 result: 10
Step 8
Back
Next
Step 9:Combine the two new left and right 2-bits:
Combination: 0010
Step 9
Step 10:Permute 4 last output:
P4 Permute: 0010
Step 10
Step 11:Last Output XOR Left half of initial cipher input:
P4 XOR IP LEFT HALF: 0100
Step 11
Step 12:Combine last output right half of initial cipher input:
Combination of previous output and ip right half: 01001001
Step 12
Step 13:Divide to 2 halfs:
Left Half: 0100 , Right Half: 1001
Step 13
Step 14:Swap Left & Right half's Positions:
Swap L-R half's: 10010100
Step 14

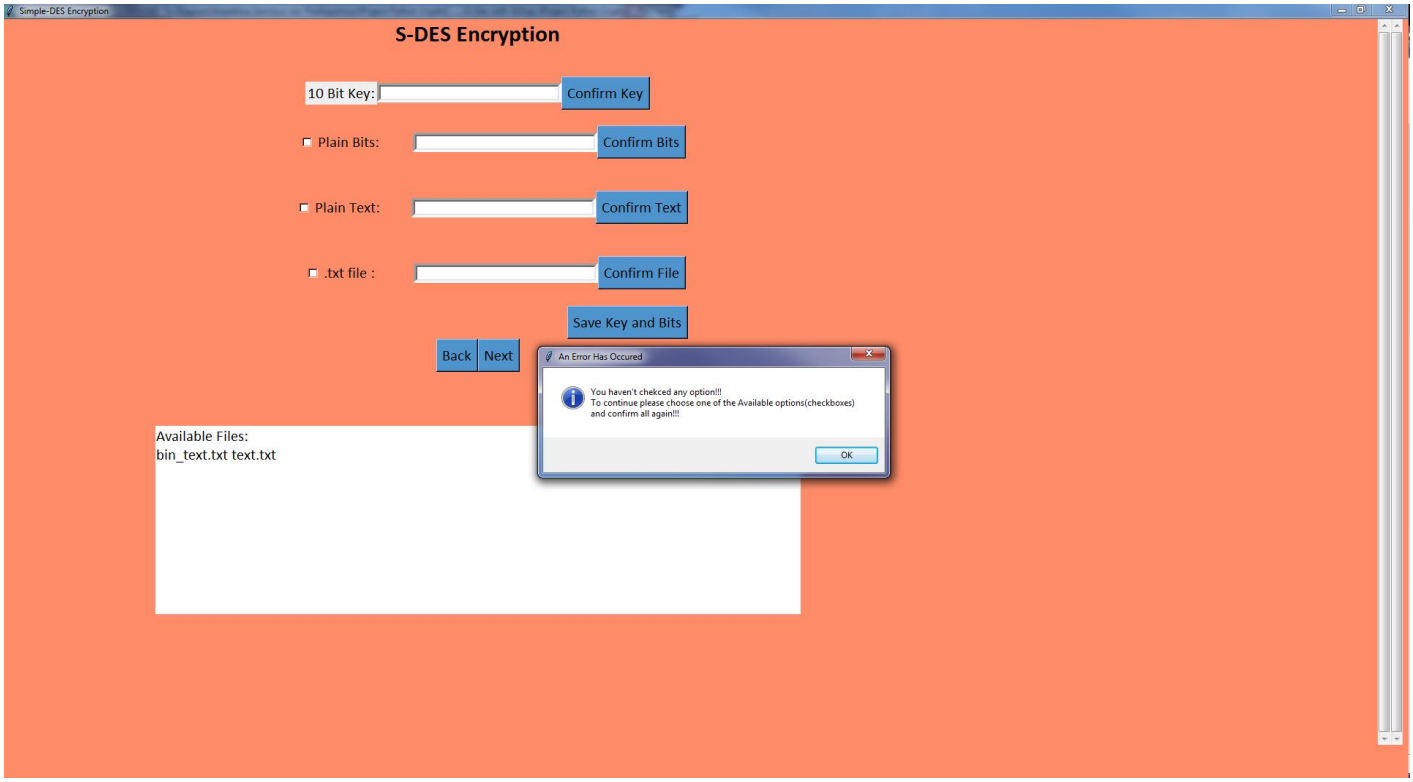

Simple-DES Encryption
S-DES Decryption
(IP^-1)IP8-inversed=[3,0,2,4,6,1,7,5], Permute4=[1,3,2,0], Expand And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of the bits sequence.
Loop 2 for key 1
Step 1:8-bit input after SW(swap):
8-bit input: 10010100
Step 1
Step 2:Divide to 2 halfs:
Left Half: 1001 , Right Half: 0100
Step 2
Step 3::Expand And Permutate right half:
EP right half: 00101000
Step 3
Step 4:Previous output XOR key 2:
EP XOR KEY 1: 00000111
Step 4
Step 5:Divide to 2 halfs:
Left Half: 0000 , Right Half: 0111
Step 5
Step 6:Left S0-box:
LEFT S0 result: 01
Step 6
Step 7:Right S1-box:
Right S1 result: 11
Step 7
Back
Next
Step 8:Combine the two new left and right 2-bits:
Combination: 0111
Step 8
Step 9:Permute 4 last output:
P4 Permute: 1110
Step 9
Step 10::Last Output XOR Left half of initial input:
P4 XOR IP LEFT HALF: 0111
Step 10
Step 11:Combine last output right half of initial input:
Combination of previous output and ip right half: 01110100
Step 11
Step 12: PermuteIP^-1 8-bit-> -> De-Ciphered text/bits:
Cipher result: 10100101
Step 12

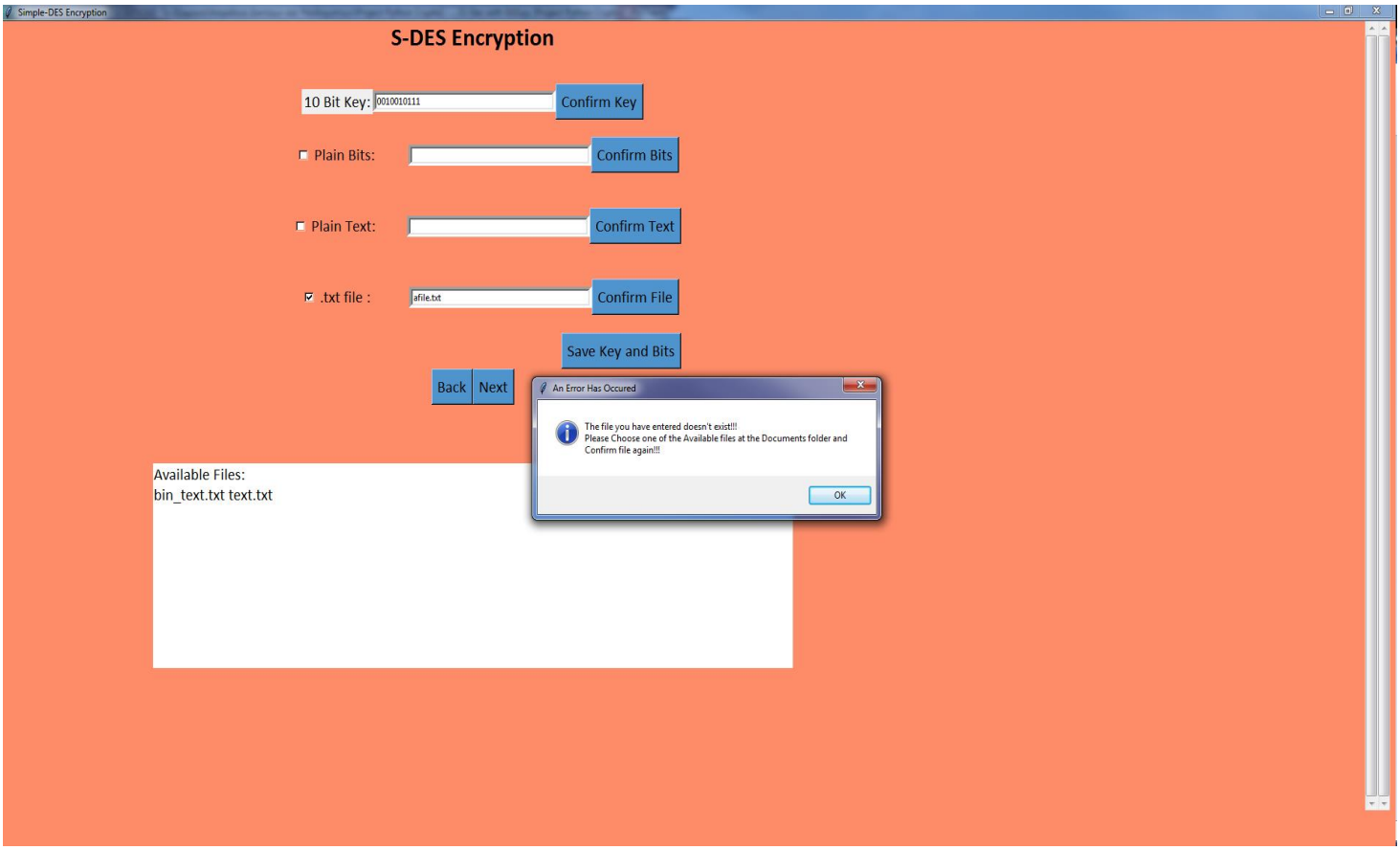
Πιθανά Σφάλματα χρήστη:




Simple-DES Encryption
S-DES Encryption
10 Bit Key:
Confirm Key
Plain Bits:
Confirm Bits
Plain Text:
Confirm Text
.txt file :
Confirm File
Save Key and Bits
An Error Has Occured
You haven't chekced any option!!!
To continue please choose one of the Available options(checkboxes)
and confirm all again!!!
OK
Available Files:
bin_text.txt text.txt

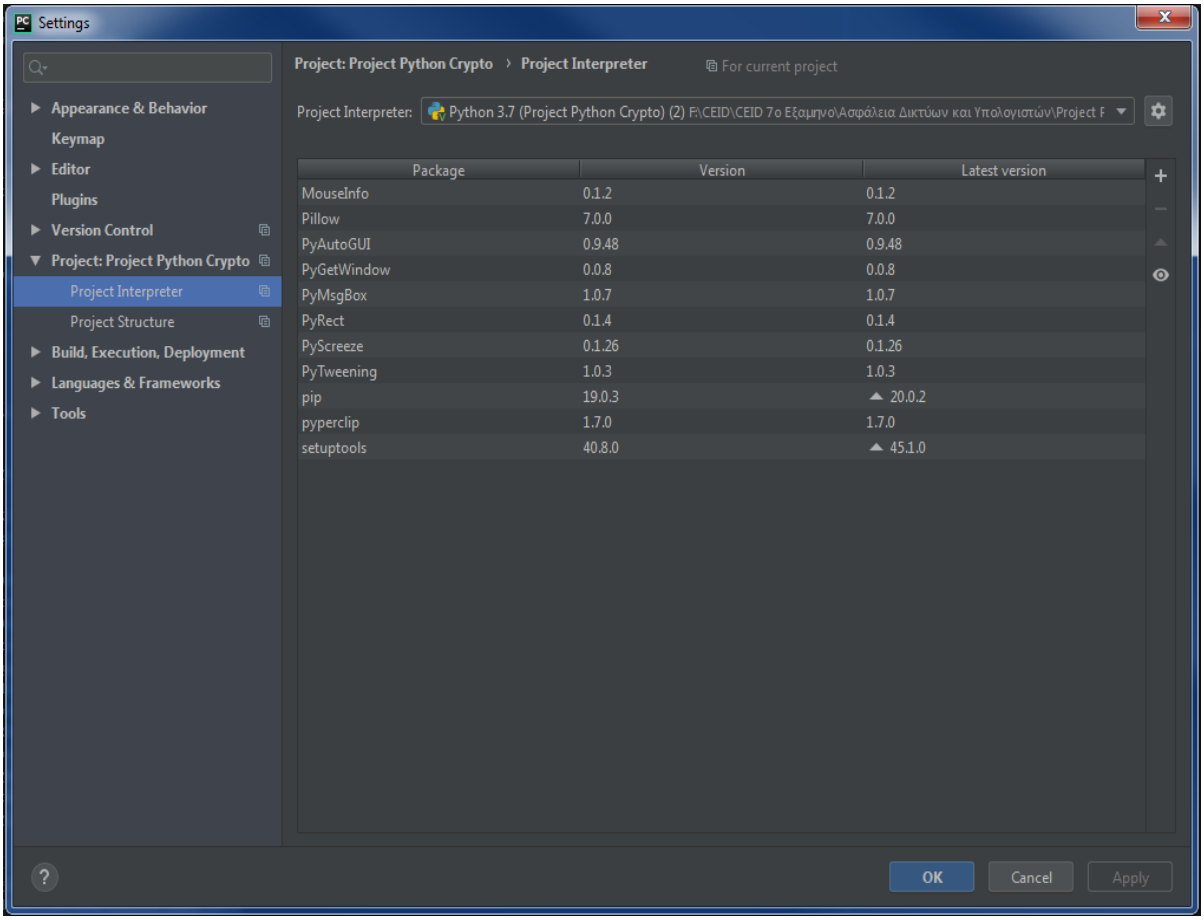

Simple-DES Encryption
S-DES Encryption
10 Bit Key:
Confirm Key
Plain Bits:
Confirm Bits
Plain Text:
Confirm Text
.txt file :
Confirm File
Save Key and Bits
Back
An Error Has Occured
No input detected!!!
Please write a suitable input and confirm again!!!
OK
Available Files:
bin_text.txt text.txt


Simple-DES Encryption
S-DES Encryption
10 Bit Key:
Confirm Key
Plain Bits:
Confirm Bits
Plain Text:
Confirm Text
.txt file :
Confirm File
Save Key and Bits
Back
Next
An Error Has Occured
You haven't chekced any option!!!
To continue please choose one of the Available options(checkboxes)
and confirm all again!!!
OK
Available Files:
bin_text.txt text.txt


Simple-DES Encryption
S-DES Encryption
10 Bit Key:
0010010111
Confirm Key
Plain Bits:
Confirm Bits
Plain Text:
Confirm Text
.txt file :
afile.txt
Confirm File
Save Key and Bits
Back
Next
An Error Has Occured
The file you have entered doesn't exist!!!
Please Choose one of the Available files at the Documents folder and
Confirm file again!!!
OK
Available Files:
bin_text.txt text.txt

Κώδικας του Προγράμματος:

Ο κώδικας για να τρέξει πρέπει ο interpreter να έχει τα εξής πακέτα.:



Ακολουθεί στην συνέχεια ο κώδικας στην python:

```
#ΒΑΖΑΙΟΣ ΣΤΥΛΙΑΝΟΣ Α.Μ.: 1054284
import os
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from PIL import Image, ImageTk
import pyautogui
from datetime import datetime


mydir=os.getcwd()
getRes=pyautogui.size()
resolution=str(getRes[0])+"x"+str(getRes[1])
key=""
plaintext=""
p10Sequence=(2,4,1,6,3,9,0,8,7,5)
p8Sequence=(5,2,6,3,7,4,9,8)
p4Sequence=(1,3,2,0)
IP8=(1,5,2,0,3,7,4,6)
IP8reversed=(3,0,2,4,6,1,7,5)
expandAndPermutate=(3,0,1,2,1,2,3,0)
S0=(("01","00","11","10"),("11","10","01","00"),("00","10","01","11"),
("11","01","11","10"))
S1=(("00","01","10","11"),("10","00","01","11"),("11","00","01","00"),
("10","01","00","11"))
p10key=[]
p8key=[]
p4key=[]
leftKey = []
rightKey = []
count=0
iteration_count=0
dec_iteration_count=0
checkBoxError=0
boxIsCheched=0
boxConfirmed=0
dummyVar=[]
text1=""
menuMessages=""
errorMessage=""
filesList=""
filesMessage=""
group8=[]
addedBits=0
kind=""
key1=""
key2=""
combKey=""
combKey2=""
ordchar=[]
keysteps=["0","0","0","0","0","0","0","0","0","0","0"]
encryptionsteps=["0","0","0","0","0","0","0","0","0","0","0","0","0","0","
0","0","0","0"]
encryptionsteps2=["0","0","0","0","0","0","0","0","0","0","0","0","0","0",
"0"]
cipher_file_name=""
decryptionsteps=["0","0","0","0","0","0","0","0","0","0","0","0","0","0","
0","0","0","0"]
decryptionsteps2=["0","0","0","0","0","0","0","0","0","0","0","0","0","0",
"0"]
```

```
def renameSpaceToUnderscore(): #this function renames the documents
that have spaces " " in their names and replaces " " with underscore
"_"
    path = os.getcwd() + "\Documents"
    filenames = os.listdir(path)
    for filename in filenames:
        os.rename(os.path.join(path, filename), os.path.join(path,
filename.replace(' ', '_')))


try:    #this try
    if not os.path.exists('./Documents'):
        os.makedirs('Documents')
        menuMessages="Folder Documents was created.\n Please restart
the programm having the Documents you want to check in the specified
Documents file\n "

    else:
        menuMessages = "Folder Documents already exists\n "
        menuMessages = menuMessages + "Please ensure that all the
documents to be checked are included in the file \"Documents\" in the
folowing directory:\n " + (mydir) + "\Documents\n "
        menuMessages = menuMessages + "Also any filenames that have
spaces, are going to have the spaces replaced by underscores(_)."
        renameSpaceToUnderscore()

    if not os.path.exists('./Encrypted Documents'):
        os.makedirs('Encrypted Documents')
        menuMessages=menuMessages+" Folder Encrypted Documents was
created\n"
    else:
        menuMessages=menuMessages+" Folder Encrypted Documents already
exists\n"

    if not os.path.exists('./Decrypted Documents'):
        os.makedirs('Decrypted Documents')
        menuMessages=menuMessages+" Folder Decrypted Documents was
created\n"
    else:
        menuMessages=menuMessages+" Folder Decrypted Documents already
exists\n"


except OSError:
    errorMessage="Error creating directory "+(mydir)+"\Documents"


allDocNum=len(os.listdir(mydir + "\Documents"))#number of documents
inside the Documents Folder

if (allDocNum == 0):
    filesMessage="There's no documents to check. Please check the
Documents file!!!"
else:
    filesMessage="The files inside the Documents folder are:\n "
    filesList=os.listdir(mydir + "\Documents")
    filesMessageText=filesMessage+str(filesList)

notifications=menuMessages+errorMessage+"\n"+filesMessage
print(notifications)
```

```
print(filesList)


def strTobin(strtext):

    global ordchar

    for e in strtext: #to know how many bits each word is.
        st1 = ord(e)
        st2 = format(st1, 'b')
        ordchar.append(len(st2))

    print("st1=",st1)
    print("st2=", st2)
    print("ordchar=", ordchar)
    str_Binary = ''.join(format(ord(i), 'b') for i in strtext)
    return str_Binary

def binTostr(bintext):  #input will be ass a string all the decrypted
bits together as one string
    global ordchar
    p=0
    bitlist = []
    charlist = []
    result_string = []
    c1=0
    while p<len(ordchar):
        l=ordchar[p]
        bit_to_add=bintext[c1:(c1+l)]
        bitlist.append(bit_to_add)
        c1=c1+l
        p=p+1
    for i in bitlist:  #save chars to each cell
        char_to_add=BintoDec(i)
        charlist.append(char_to_add)
    for i in charlist:
        res_to_add=chr(i)
        result_string.append(res_to_add)

    bin_str=''.join(result_string)

    return bin_str

def addTill8bits(strtext,fillbits):
    global addedBits
    addedBits=fillbits
    z=0
    while z<fillbits:
        strtext=strtext+"0"
        z=z+1
    return strtext




def save_text(entry):
    global text1
    global group8
    global mydir
    global kind
    global key1
```

```
    global key2
    global cipher_file_name


    global plaintext
    text1 = entry.get()
    print("entry text=",text1)

    if dummyVar[0].get()==1:
        kind="binary"
    elif dummyVar[1].get()==1:
        kind="text"
    elif dummyVar[2].get()==1:
        kind="file"
    else:
        kind="NaN"
        unchecked_option()
    print("kind=",kind)

    if kind=="binary":
        if len(text1) == 0:
            no_inputError()
        if len(text1) == 8:
            plaintext=text1
        elif len(text1) < 8:
            add=8-len(text1)
            text1 = addTill8bits(text1,add)
            plaintext = text1
        elif(len(text1) % 8 == 0):
            plaintext = text1
        else:
            modulo=len(text1) % 8
            add=8-modulo
            text1 = addTill8bits(text1, add)
            plaintext = text1
        print("addedBitplaintext=",plaintext)

    elif kind=="text":
        if len(text1) == 0:
            no_inputError()
        text1 = strTobin(text1)
        if len(text1)==8:
            plaintext=text1
        elif len(text1)<8:
            add=8-len(text1)
            text1 = addTill8bits(text1,add)
            plaintext = text1
        elif (len(text1) % 8 == 0):
            plaintext = text1
        else:
            modulo=len(text1) % 8
            add=8-modulo
            text1 = addTill8bits(text1, add)
            plaintext = text1
        print("addedBitplaintext2=",plaintext)

    elif kind=="file":
        if len(text1) == 0:
            no_inputError()
        cipher_file_name=text1
        filepath = mydir + "\\Documents" + "\\" + text1
```

```
        if not os.path.exists(filepath):
            fileError()
        file = open(filepath, "r")
        string1=file.read()
        print("string1=",string1)
        string1 = strTobin(string1)
        print("string1BIN=", string1)
        if len(string1)==8:
            plaintext=string1
        elif len(string1)<8:
            add=8-len(string1)
            text1 = addTill8bits(string1,add)
            plaintext = string1
        elif (len(string1) % 8 == 0):
            plaintext = string1
        else:
            modulo=len(string1) % 8
            add=8-modulo
            string1 = addTill8bits(string1, add)
            print("string1till=", string1)
            plaintext = string1
        print("addedBitplaintext3=",plaintext)



def inputError():
    messagebox.showinfo("An Error Has Occured", "The input key must be
10-bit long!!!\nPlease write a suitable 10-bit key and confirm key
again!!!")


def fileError():
    messagebox.showinfo("An Error Has Occured", "The file you have
entered doesn't exist!!!\nPlease Choose one of the Available files at
the Documents folder and Confirm file again!!!")

def no_inputError():
    messagebox.showinfo("An Error Has Occured","No input
detected!!!\nPlease write a suitable input and confirm again!!!")

def unchecked_option():
    messagebox.showinfo("An Error Has Occured","You haven't chekced
any option!!!\nTo continue please choose one of the Available
options(checkboxes) and confirm all again!!!")




def set_CheckBoxVar():
    global boxIsCheched
    boxIsCheched=1
    print("set_CheckBoxVar",boxIsCheched)

def boxConfirmation():
    global boxConfirmed
    if boxIsCheched==1:
        boxConfirmed=1
    else:
        boxConfirmed = 0
```

```
def get_inputkey(Entry):
    global key
    if len(Entry.get())!=10:
        inputError()
    key=Entry.get()
    print(key)


def raise_frame(frame):
    frame.tkraise()
    frame.grid(row=int(getRes[0]),column=0,sticky="news")


def main_Encrypt_window():

    global key
    global checkBoxError
    global plaintext
    global boxIsCheched
    global text1
    global dummyVar
    global leftKey
    global rightKey
    global p10key
    global p8key
    global p4key
    global combKey
    global combKey2
    global keysteps



    main_window=Tk()
    main_window.geometry(resolution)
##########################################resolution
    main_window.title("Simple-DES Encryption")
    main_window.configure(background='salmon1')
    #--------------------------------------------------
    container = ttk.Frame(main_window,width=getRes[0],
height=getRes[1])
    canvas = Canvas(container, bg="salmon1",width=getRes[0]-50,
height=getRes[1]-70,borderwidth=0,highlightthickness=0)
    scrollbarv = ttk.Scrollbar(container, orient="vertical",
command=canvas.yview)
    scrollbarh = ttk.Scrollbar(container, orient="vertical",
command=canvas.xview)
    scrollable_frame = ttk.Frame(canvas)

    scrollable_frame.bind("<Configure>",lambda e:
canvas.configure(scrollregion=canvas.bbox("all")))

    canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")

    canvas.configure(yscrollcommand=scrollbarv.set,
xscrollcommand=scrollbarh.set)

    #--------------------------------------------------


    introFrame=Frame(scrollable_frame, bg="salmon1",width=getRes[0],
```

```
height=getRes[1])
    choosesdesFrame=Frame(scrollable_frame,width=getRes[0],
height=getRes[1], bg="salmon1")
    sdesFrame1=Frame(scrollable_frame,width=getRes[0],
height=getRes[0], bg="salmon1")
    sdesFrame2=Frame(scrollable_frame, bg="salmon1",width=getRes[0],
height=getRes[1])
    sdesFrame3 = Frame(scrollable_frame, bg="salmon1",
width=getRes[0], height=getRes[1])
    sdesFrame4 = Frame(scrollable_frame, bg="salmon1",
width=getRes[0], height=getRes[1])
    sdesFrame5 = Frame(scrollable_frame, bg="salmon1",
width=getRes[0], height=getRes[1])
    sdesFrame6 = Frame(scrollable_frame, bg="salmon1",
width=getRes[0], height=getRes[1])




    load1=Image.open('sdesKeys.png')
    render1=ImageTk.PhotoImage(load1)
    load2 = Image.open('sdesEnc.png')
    render2=ImageTk.PhotoImage(load2)
    load3 = Image.open('sdesDec.png')
    render3 = ImageTk.PhotoImage(load3)
    topTitle=Label(introFrame,borderwidth=0,highlightthickness=0,
bg="salmon1")
    L1=Label(topTitle,text="Simplified-DES Encryption\Decryption\n",
bg="salmon1",font=("Calibri", 24,"bold"))
    topLabel2=Label(introFrame,borderwidth=0,highlightthickness=0,
bg="salmon1")

descriptionText=Text(topLabel2,height=7,bg="salmon1",borderwidth=0,hig
hlightthickness=0,font=("Calibri", 14), width = 100)
    descriptionText.insert(INSERT,"Περιγραφή S-DES:\nO αλγόριθμος
κρυπτογράφησης Simplified Data Encryption Standard (S-DES) βασίζεται
στην κρυπτογράφηση μπλοκ.\n Παίρνει σαν είσοδο ένα μπλοκ 8-bit απλού
κειμένου ή 8-bit και ένα κλειδί μεγέθους  10-bit και τελικά παράγει
ένα\n κρυπτογραφημένο μπλοκ 8-bit ciphertext ως έξοδο.Πρόκειται για
ένα συμμετρικό κρυπτογράφο που έχει δύο επαναλήψεις,\n ενώ για την
κρυπτογράφηση\αποκρυπτογράφηση χρησιμοποιούνται συγκεκριμένες
μεταθέσεις των bits που ορίζονται από τον ίδιο τον αλγόριθμο.\n")


butttonNext1=Button(topLabel2,text="Next",command=lambda:raise_frame(c
hoosesdesFrame),bg="SteelBlue3",height = 2, width = 7)

buttonExit1=Button(topLabel2,text="Exit",command=lambda:main_window.de
stroy,bg="SteelBlue3",height = 2, width = 5)

    bottomLabel=Label(introFrame,borderwidth=0,highlightthickness=0,
bg="salmon1")
    P1 =
Label(bottomLabel,image=render1,borderwidth=0,highlightthickness=0)
    P2 =
Label(bottomLabel,image=render2,borderwidth=0,highlightthickness=0)
    P3 =
Label(bottomLabel,image=render3,borderwidth=0,highlightthickness=0)
    #P3 =
Label(bottomLabel,padx=(getRes[0]/10),bg="salmon1",borderwidth=0,highl
ightthickness=0)
```

```
    #P4 =
Label(bottomLabel,image=render3,borderwidth=0,highlightthickness=0)
    L1.pack(side=TOP)
    topTitle.pack(side=TOP)
    topLabel2.pack(side=TOP)
    descriptionText.pack()
    butttonNext1.pack(side=RIGHT)
    buttonExit1.pack(side=LEFT)
    P1.pack(side=LEFT)
    P2.pack(side=LEFT)
    P3.pack(side=RIGHT)
    bottomLabel.pack(side=BOTTOM)
    raise_frame(introFrame)
    #-------------------------------------------------------------------
--------------------------------First Frame END, Start of PAGE 1

    mainTop2 = Label(choosesdesFrame, borderwidth=0,
highlightthickness=0, bg="salmon1")
    topTitle2 = Label(mainTop2, borderwidth=0, highlightthickness=0,
text="S-DES Encryption\n",bg="salmon1", font=("Calibri", 24,"bold"))
    insidekeyLabel=Label(mainTop2, borderwidth=0,
highlightthickness=0,bg="salmon1")
    keyLabel=Label(insidekeyLabel,text="10 Bit Key:",font=("Calibri",
16))
    keyEntry = Entry(insidekeyLabel, bd=5, width=40)
    buttonKeyConfirm = Button(insidekeyLabel, text="Confirm Key",
command=lambda:get_inputkey(keyEntry), bg="SteelBlue3",
font=("Calibri", 16))############################SOS IF KEY< OR >10
BITS ERROR
    insideText = Label(mainTop2, borderwidth=0, highlightthickness=0,
bg="salmon1")
    insideText1=Label(insideText, borderwidth=0,
highlightthickness=0,bg="salmon1")
    insideText2 = Label(insideText, borderwidth=0,
highlightthickness=0, bg="salmon1")
    insideText3 = Label(insideText, borderwidth=0,
highlightthickness=0, bg="salmon1")
    exit_next = Label(choosesdesFrame, borderwidth=0,
highlightthickness=0, bg="salmon1")
    z=0
    while z<1:
        dummyVar.append(IntVar())
        dummyVar.append(IntVar())
        dummyVar.append(IntVar())
        C1 = Checkbutton(insideText1, text="Plain
Bits:",variable=dummyVar[z], command=lambda:set_CheckBoxVar(),
font=("Calibri",
16),height=3,width=15,bg="salmon1",activebackground="SteelBlue3")
        C2 = Checkbutton(insideText2, text="Plain
Text:",variable=dummyVar[z+1], command=lambda:set_CheckBoxVar(),
font=("Calibri",
16),height=3,width=15,bg="salmon1",activebackground="SteelBlue3")
        C3 = Checkbutton(insideText3, text=".txt file
:",variable=dummyVar[z+2], command=lambda:set_CheckBoxVar(),
font=("Calibri",
16),height=3,width=15,bg="salmon1",activebackground="SteelBlue3")
        z=z+1
```

```
    textEntry1 = Entry(insideText1, bd=5, width=40)
    textEntry2 = Entry(insideText2, bd=5, width=40)
    textEntry3 = Entry(insideText3, bd=5, width=40)

    buttonTextConfirm1 = Button(insideText1, text="Confirm Bits",
command=lambda: save_text(textEntry1),bg="SteelBlue3",
font=("Calibri", 16))
    buttonTextConfirm2 = Button(insideText2, text="Confirm Text",
command=lambda: save_text(textEntry2),bg="SteelBlue3",
font=("Calibri", 16))
    buttonTextConfirm3 = Button(insideText3, text="Confirm File",
command=lambda: save_text(textEntry3),bg="SteelBlue3",
font=("Calibri", 16))
    buttonTextConfirm = Button(insideText, text="Save Key and
Bits",command=lambda:main(),  bg="SteelBlue3", font=("Calibri", 16))


    buttonToExit2 = Button(exit_next, text="Back",
command=lambda:raise_frame(introFrame),bg="SteelBlue3",
font=("Calibri", 16))
    buttonToContinue2 = Button(exit_next,
text="Next",command=lambda:raise_frame(sdesFrame1),
bg="SteelBlue3",font=("Calibri", 16))
    NotificationsLogText = Text(choosesdesFrame,height=10,
bg="white", borderwidth=0, highlightthickness=0, font=("Calibri", 16))
    NotificationsLogText.insert(INSERT, "Available
Files:\n",notifications,filesList)
    NotificationsLogText.config(state=DISABLED)


    mainTop2.pack(side=TOP)
    topTitle2.pack(side=TOP)
    insidekeyLabel.pack(side=TOP)
    keyLabel.pack(side=LEFT)
    buttonKeyConfirm.pack(side=RIGHT)
    keyEntry.pack(side=RIGHT)
    insideText.pack(side=TOP)
    insideText1.pack(side=TOP)
    insideText2.pack(side=TOP)
    insideText3.pack(side=TOP)
    exit_next.pack(side=TOP)
    C1.pack(side=LEFT)
    C2.pack(side=LEFT)
    C3.pack(side=LEFT)
    buttonTextConfirm1.pack(side=RIGHT)
    buttonTextConfirm2.pack(side=RIGHT)
    buttonTextConfirm3.pack(side=RIGHT)
    buttonToExit2.pack(side=LEFT)
    buttonToContinue2.pack(side=RIGHT)
    textEntry1.pack(side=RIGHT)
    textEntry2.pack(side=RIGHT)
    textEntry3.pack(side=RIGHT)
    buttonTextConfirm.pack(side=RIGHT)
    NotificationsLogText.pack(side=BOTTOM)

   #-----------------------------------------------------------------
----------------------------------------------------------PAGE 3 SDES
KEY GENERATING
    # -----------------------------------------------------------------
----------------------------------------------------------PAGE 3
SDES KEY GENERATING
```

```
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 3
SDES KEY GENERATING
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 3
SDES KEY GENERATING
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 3
SDES KEY GENERATING
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 3
SDES KEY GENERATING

    topTitle3 = Label(sdesFrame1, borderwidth=0, highlightthickness=0,
text="S-DES Key Generating\n", bg="salmon1",font=("Calibri",
24,"bold"))


    bigkeyLabel1 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel2 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel3 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel4 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel5 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel6 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel7 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel8 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel9 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel10 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel11 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel12 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0,bg="salmon1")
    bigkeyLabel13 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigkeyLabel14 = Label(sdesFrame1, borderwidth=0,
highlightthickness=0, bg="salmon1")

    keyStep0  = Label(sdesFrame1, borderwidth=0, highlightthickness=0,
text="    Permute10=[2,4,1,6,3,9,0,8,7,5], Permute8=[5,2,6,3,7,4,9,8],
where 0 to 1 is the element position of the initial 10-bit key.
\n",bg="salmon1", font=("Calibri", 17))
    keyStep1  = Label(bigkeyLabel1, borderwidth=0,
highlightthickness=0, text="Step 1:Permute the initial 10-bit key:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep2  = Label(bigkeyLabel2, borderwidth=0,
highlightthickness=0, text="Step 2:Divide key to 2 halfs:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    keyStep3  = Label(bigkeyLabel3, borderwidth=0,
highlightthickness=0, text="Step 3:Left Round Shift by 1 the left
half:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep4  = Label(bigkeyLabel4, borderwidth=0,
```

```
highlightthickness=0, text="Step 4:Right Round Shift by 1 the right
half:", bg="salmon1", font=("Calibri", 17),height=2,width=50)
    keyStep5  = Label(bigkeyLabel5, borderwidth=0,
highlightthickness=0, text="Step 5:Combine Left and right half after
each shift:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep6  = Label(bigkeyLabel6, borderwidth=0,
highlightthickness=0, text="Step 6:Permute 8bit key",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep7  = Label(bigkeyLabel7, borderwidth=0,
highlightthickness=0, text="Key 1:", bg="salmon1",font=("Calibri",
17,"bold"),height=2,width=50)
    keyStep8  = Label(bigkeyLabel8, borderwidth=0,
highlightthickness=0, text="Step 7:Left Round Shift by 1 the left half
again:    ", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep9  = Label(bigkeyLabel9, borderwidth=0,
highlightthickness=0, text="Step 8:Left Round Shift by 1 the left half
again:    ", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep10 = Label(bigkeyLabel10, borderwidth=0,
highlightthickness=0, text="Step 9:Right Round Shift by 1 the right
half again:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep11 = Label(bigkeyLabel11, borderwidth=0,
highlightthickness=0, text="Step 10:Right Round Shift by 1 the right
half again:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep12 = Label(bigkeyLabel12, borderwidth=0,
highlightthickness=0, text="Step 11:Combine the two new left and right
halfs:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep13 = Label(bigkeyLabel13, borderwidth=0,
highlightthickness=0, text="Step 12:Permute 8bit key:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    keyStep14 = Label(bigkeyLabel14, borderwidth=0,
highlightthickness=0, text="Key 2:", bg="salmon1",font=("Calibri",
17,"bold"),height=2,width=50)

    var1 = StringVar()
    var1.set("")
    var2 = StringVar()
    var2.set("")
    var3 = StringVar()
    var3.set("")
    var4 = StringVar()
    var4.set("")
    var5 = StringVar()
    var5.set("")
    var6 = StringVar()
    var6.set("")
    var7 = StringVar()
    var7.set("")
    var8 = StringVar()
    var8.set("")
    var9 = StringVar()
    var9.set("")
    var10 = StringVar()
    var10.set("")
    var11 = StringVar()
    var11.set("")
    var12 = StringVar()
    var12.set("")
    var13 = StringVar()
    var13.set("")
    var14 = StringVar()
    var14.set("")
```

```
    def call_s1():
        var1.set("P10: "+''.join(keysteps[0]))

    def call_s2():
        var2.set("Left Half: " + ''.join(keysteps[1])+" ,Right Half:
"+''.join(keysteps[2]))

    def call_s3():
        var3.set("Left key: "+''.join(keysteps[3]))

    def call_s4():
        var4.set("Right key: " + ''.join(keysteps[4]))

    def call_s5():
        var5.set("Combination of keys: "+''.join(keysteps[5]))

    def call_s6():
        var6.set("P8: "+''.join(key1))

    def call_s7():
        var7.set("Key 1: "+''.join(key1))

    def call_s8():
        var8.set("Left key: "+''.join(keysteps[6]))

    def call_s9():
        var9.set("Left key: "+''.join(keysteps[7]))

    def call_s10():
        var10.set("Right key: " + ''.join(keysteps[8]))

    def call_s11():
        var11.set("Right key: " + ''.join(keysteps[9]))

    def call_s12():
        var12.set("Combination of keys: "+''.join(keysteps[10]))

    def call_s13():
        var13.set("P8: "+''.join(key2))

    def call_s14():
        var14.set("Key 2: "+''.join(key2))



    keyStepText1 = Label(bigkeyLabel1, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var1)
    keyStepText2 = Label(bigkeyLabel2, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var2)
    keyStepText3 = Label(bigkeyLabel3, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var3)
    keyStepText4 = Label(bigkeyLabel4, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var4)
    keyStepText5 = Label(bigkeyLabel5, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var5)
    keyStepText6 = Label(bigkeyLabel6, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var6)
    keyStepText7 = Label(bigkeyLabel7, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var7)
    keyStepText8 = Label(bigkeyLabel8, bg="white", borderwidth=0,
```

```
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var8)
    keyStepText9 = Label(bigkeyLabel9, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var9)
    keyStepText10 = Label(bigkeyLabel10, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var10)
    keyStepText11 = Label(bigkeyLabel11, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var11)
    keyStepText12 = Label(bigkeyLabel12, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var12)
    keyStepText13 = Label(bigkeyLabel13, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var13)
    keyStepText14 = Label(bigkeyLabel14, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=var14)



    butttonToNext3=Button(sdesFrame1,text="Next",command=lambda:
raise_frame(sdesFrame2),bg="SteelBlue3",height = 3, width = 8)

buttonToExit3=Button(sdesFrame1,text="Back",command=lambda:raise_frame
(choosesdesFrame),bg="SteelBlue3",height = 3, width = 8)
    b1 = Button(bigkeyLabel1, text="Step
1",command=lambda:call_s1(),bg="SteelBlue3",height = 1, width = 6)
    b2 = Button(bigkeyLabel2, text="Step
2",command=lambda:call_s2(),bg="SteelBlue3",height = 1, width = 6)
    b3 = Button(bigkeyLabel3, text="Step
3",command=lambda:call_s3(),bg="SteelBlue3",height = 1, width = 6)
    b4 = Button(bigkeyLabel4, text="Step
4",command=lambda:call_s4(),bg="SteelBlue3",height = 1, width = 6)
    b5 = Button(bigkeyLabel5, text="Step
5",command=lambda:call_s5(),bg="SteelBlue3",height = 1, width = 6)
    b6 = Button(bigkeyLabel6,text="Step
6",command=lambda:call_s6(),bg="SteelBlue3",height = 1, width = 6)
    b7 = Button(bigkeyLabel7, text="Step
7",command=lambda:call_s7(),bg="SteelBlue3",height = 1, width = 6)
    b8 = Button(bigkeyLabel8, text="Step
8",command=lambda:call_s8(),bg="SteelBlue3",height = 1, width = 6)
    b9 = Button(bigkeyLabel9,text="Step
9",command=lambda:call_s9(),bg="SteelBlue3",height = 1, width = 6)
    b10 = Button(bigkeyLabel10, text="Step
10",command=lambda:call_s10(),bg="SteelBlue3",height = 1, width = 6)
    b11 = Button(bigkeyLabel11, text="Step
11",command=lambda:call_s11(),bg="SteelBlue3",height = 1, width = 6)
    b12= Button(bigkeyLabel12, text="Step
12",command=lambda:call_s12(),bg="SteelBlue3",height = 1, width = 6)
    b13 = Button(bigkeyLabel13, text="Step
13",command=lambda:call_s13(),bg="SteelBlue3",height = 1, width = 6)
    b14 = Button(bigkeyLabel14, text="Step
14",command=lambda:call_s14(),bg="SteelBlue3",height = 1, width = 6)

    topTitle3.pack(side=TOP)
    keyStep0.pack(side=TOP)

    keyStep1.pack(side=LEFT)
    keyStep2.pack(side=LEFT)
    keyStep3.pack(side=LEFT)
    keyStep4.pack(side=LEFT)
    keyStep5.pack(side=LEFT)
    keyStep6.pack(side=LEFT)
    keyStep7.pack(side=LEFT)
    keyStep8.pack(side=LEFT)
```

```
    keyStep9.pack(side=LEFT)
    keyStep10.pack(side=LEFT)
    keyStep11.pack(side=LEFT)
    keyStep12.pack(side=LEFT)
    keyStep13.pack(side=LEFT)
    keyStep14.pack(side=LEFT)

    b1.pack(side=RIGHT)
    b2.pack(side=RIGHT)
    b3.pack(side=RIGHT)
    b4.pack(side=RIGHT)
    b5.pack(side=RIGHT)
    b6.pack(side=RIGHT)
    b7.pack(side=RIGHT)
    b8.pack(side=RIGHT)
    b9.pack(side=RIGHT)
    b10.pack(side=RIGHT)
    b11.pack(side=RIGHT)
    b12.pack(side=RIGHT)
    b13.pack(side=RIGHT)
    b14.pack(side=RIGHT)

    keyStepText1.pack(side=RIGHT)
    keyStepText2.pack(side=RIGHT)
    keyStepText3.pack(side=RIGHT)
    keyStepText4.pack(side=RIGHT)
    keyStepText5.pack(side=RIGHT)
    keyStepText6.pack(side=RIGHT)
    keyStepText7.pack(side=RIGHT)
    keyStepText8.pack(side=RIGHT)
    keyStepText9.pack(side=RIGHT)
    keyStepText10.pack(side=RIGHT)
    keyStepText11.pack(side=RIGHT)
    keyStepText12.pack(side=RIGHT)
    keyStepText13.pack(side=RIGHT)
    keyStepText14.pack(side=RIGHT)

    butttonToNext3.pack(side=RIGHT)
    buttonToExit3.pack(side=RIGHT)

    bigkeyLabel1.pack(side=TOP)
    bigkeyLabel2.pack(side=TOP)
    bigkeyLabel3.pack(side=TOP)
    bigkeyLabel4.pack(side=TOP)
    bigkeyLabel5.pack(side=TOP)
    bigkeyLabel6.pack(side=TOP)
    bigkeyLabel7.pack(side=TOP)
    bigkeyLabel8.pack(side=TOP)
    bigkeyLabel9.pack(side=TOP)
    bigkeyLabel10.pack(side=TOP)
    bigkeyLabel11.pack(side=TOP)
    bigkeyLabel12.pack(side=TOP)
    bigkeyLabel13.pack(side=TOP)
    bigkeyLabel14.pack(side=TOP)

    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 4
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 4
SDES ENCRYPTION
```

```
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 4
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 4
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 4
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 4
SDES ENCRYPTION


    topTitle3 = Label(sdesFrame2, borderwidth=0, highlightthickness=0,
text="S-DES Encryption\n", bg="salmon1",font=("Calibri", 24,"bold"))


    bigEncLabel1 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel2 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel3 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel4 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel5 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel6 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel7 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel8 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel9 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel10 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel11 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel12 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0,bg="salmon1")
    bigEncLabel13 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEncLabel14 = Label(sdesFrame2, borderwidth=0,
highlightthickness=0, bg="salmon1")

    EncStep0  = Label(sdesFrame2, borderwidth=0, highlightthickness=0,
text="          Permute8=[5,2,6,3,7,4,9,8], Permute4=[1,3,2,0], Expand
And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position
of the bits sequence. \n\nLoop 1 for Key 1",bg="salmon1",
font=("Calibri", 17))
    EncStep1  = Label(bigEncLabel1, borderwidth=0,
highlightthickness=0, text="Step 1:Initial 8-bit input:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep2  = Label(bigEncLabel2, borderwidth=0,
highlightthickness=0, text="Step 2:Permute the initial 8-bit input:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    EncStep3  = Label(bigEncLabel3, borderwidth=0,
highlightthickness=0, text="Step 3:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
```

```
    EncStep4  = Label(bigEncLabel4, borderwidth=0,
highlightthickness=0, text="Step 4:Expand And Permutate right half:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    EncStep5  = Label(bigEncLabel5, borderwidth=0,
highlightthickness=0, text="Step 5:Previous output XOR key 1:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep6  = Label(bigEncLabel6, borderwidth=0,
highlightthickness=0, text="Step 6:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep7  = Label(bigEncLabel7, borderwidth=0,
highlightthickness=0, text="Step 7:Left S0-box:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep8  = Label(bigEncLabel8, borderwidth=0,
highlightthickness=0, text="Step 8:Right S1-box:    ",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep9  = Label(bigEncLabel9, borderwidth=0,
highlightthickness=0, text="Step 9:Combine the two new left and right
2-bits:    ", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep10 = Label(bigEncLabel10, borderwidth=0,
highlightthickness=0, text="Step 10:Permute 4 last output:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep11 = Label(bigEncLabel11, borderwidth=0,
highlightthickness=0, text="Step 11:Last Output XOR Left half of
initial input:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep12 = Label(bigEncLabel12, borderwidth=0,
highlightthickness=0, text="Step 12:Combine last output  right half of
initial input:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep13 = Label(bigEncLabel13, borderwidth=0,
highlightthickness=0, text="Step 13:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    EncStep14 = Label(bigEncLabel14, borderwidth=0,
highlightthickness=0, text="Step 14:Swap Left & Right half's
Positions:", bg="salmon1",font=("Calibri", 17),height=2,width=50)

    varE1 = StringVar()
    varE1.set("")
    varE2 = StringVar()
    varE2.set("")
    varE3 = StringVar()
    varE3.set("")
    varE4 = StringVar()
    varE4.set("")
    varE5 = StringVar()
    varE5.set("")
    varE6 = StringVar()
    varE6.set("")
    varE7 = StringVar()
    varE7.set("")
    varE8 = StringVar()
    varE8.set("")
    varE9 = StringVar()
    varE9.set("")
    varE10 = StringVar()
    varE10.set("")
    varE11 = StringVar()
    varE11.set("")
    varE12 = StringVar()
    varE12.set("")
    varE13 = StringVar()
    varE13.set("")
    varE14 = StringVar()
```

```
    varE14.set("")

    def call_Es1():
        varE1.set("8-bit input: "+''.join(encryptionsteps[0]))

    def call_Es2():
        varE2.set("IP8 Permute: " + ''.join(encryptionsteps[1]))

    def call_Es3():
        varE3.set("Left Half: " + ''.join(encryptionsteps[2])+" ,
Right Half: "+''.join(encryptionsteps[3]))

    def call_Es4():
        varE4.set("EP right half: " + ''.join(encryptionsteps[4]))

    def call_Es5():
        varE5.set("EP XOR KEY 1: "+''.join(encryptionsteps[5]))

    def call_Es6():
        varE6.set("Left Half: " + ''.join(encryptionsteps[6])+" ,
Right Half: "+''.join(encryptionsteps[7]))

    def call_Es7():
        varE7.set("LEFT S0 result: "+''.join(encryptionsteps[8]))

    def call_Es8():
        varE8.set("Right S1 result: "+''.join(encryptionsteps[9]))

    def call_Es9():
        varE9.set("Combination: "+''.join(encryptionsteps[10]))

    def call_Es10():
        varE10.set("P4 Permute: " + ''.join(encryptionsteps[11]))

    def call_Es11():
        varE11.set("P4 XOR  IP LEFT HALF: " +
''.join(encryptionsteps[12]))

    def call_Es12():
        varE12.set("Combination of previous output and ip right half:
"+ ''.join(encryptionsteps[13]))

    def call_Es13():
        varE13.set("Left Half: " + ''.join(encryptionsteps[14])+" ,
Right Half: "+''.join(encryptionsteps[15]))

    def call_Es14():
        varE14.set("Swap L-R half's: "+''.join(encryptionsteps[16]))



    encStepText1 = Label(bigEncLabel1, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE1)
    encStepText2 = Label(bigEncLabel2, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE2)
    encStepText3 = Label(bigEncLabel3, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE3)
    encStepText4 = Label(bigEncLabel4, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE4)
    encStepText5 = Label(bigEncLabel5, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE5)
```

```
    encStepText6 = Label(bigEncLabel6, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE6)
    encStepText7 = Label(bigEncLabel7, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE7)
    encStepText8 = Label(bigEncLabel8, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE8)
    encStepText9 = Label(bigEncLabel9, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri", 16),height=1,textvariable=varE9)
    encStepText10 = Label(bigEncLabel10, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varE10)
    encStepText11 = Label(bigEncLabel11, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varE11)
    encStepText12 = Label(bigEncLabel12, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varE12)
    encStepText13 = Label(bigEncLabel13, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varE13)
    encStepText14 = Label(bigEncLabel14, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varE14)



    butttonToNext4=Button(sdesFrame2,text="Next",command=lambda:
raise_frame(sdesFrame3),bg="SteelBlue3",height = 3, width = 8)

buttonToExit4=Button(sdesFrame2,text="Back",command=lambda:raise_frame
(sdesFrame1),bg="SteelBlue3",height = 3, width = 8)
    bE1 = Button(bigEncLabel1, text="Step
1",command=lambda:call_Es1(),bg="SteelBlue3",height = 1, width = 6)
    bE2 = Button(bigEncLabel2, text="Step
2",command=lambda:call_Es2(),bg="SteelBlue3",height = 1, width = 6)
    bE3 = Button(bigEncLabel3, text="Step
3",command=lambda:call_Es3(),bg="SteelBlue3",height = 1, width = 6)
    bE4 = Button(bigEncLabel4, text="Step
4",command=lambda:call_Es4(),bg="SteelBlue3",height = 1, width = 6)
    bE5 = Button(bigEncLabel5, text="Step
5",command=lambda:call_Es5(),bg="SteelBlue3",height = 1, width = 6)
    bE6 = Button(bigEncLabel6,text="Step
6",command=lambda:call_Es6(),bg="SteelBlue3",height = 1, width = 6)
    bE7 = Button(bigEncLabel7, text="Step
7",command=lambda:call_Es7(),bg="SteelBlue3",height = 1, width = 6)
    bE8 = Button(bigEncLabel8, text="Step
8",command=lambda:call_Es8(),bg="SteelBlue3",height = 1, width = 6)
    bE9 = Button(bigEncLabel9,text="Step
9",command=lambda:call_Es9(),bg="SteelBlue3",height = 1, width = 6)
    bE10 = Button(bigEncLabel10, text="Step
10",command=lambda:call_Es10(),bg="SteelBlue3",height = 1, width = 6)
    bE11 = Button(bigEncLabel11, text="Step
11",command=lambda:call_Es11(),bg="SteelBlue3",height = 1, width = 6)
    bE12= Button(bigEncLabel12, text="Step
12",command=lambda:call_Es12(),bg="SteelBlue3",height = 1, width = 6)
    bE13 = Button(bigEncLabel13, text="Step
13",command=lambda:call_Es13(),bg="SteelBlue3",height = 1, width = 6)
    bE14 = Button(bigEncLabel14, text="Step
14",command=lambda:call_Es14(),bg="SteelBlue3",height = 1, width = 6)

    topTitle3.pack(side=TOP)
```

```
    EncStep0.pack(side=TOP)

    EncStep1.pack(side=LEFT)
    EncStep2.pack(side=LEFT)
    EncStep3.pack(side=LEFT)
    EncStep4.pack(side=LEFT)
    EncStep5.pack(side=LEFT)
    EncStep6.pack(side=LEFT)
    EncStep7.pack(side=LEFT)
    EncStep8.pack(side=LEFT)
    EncStep9.pack(side=LEFT)
    EncStep10.pack(side=LEFT)
    EncStep11.pack(side=LEFT)
    EncStep12.pack(side=LEFT)
    EncStep13.pack(side=LEFT)
    EncStep14.pack(side=LEFT)

    bE1.pack(side=RIGHT)
    bE2.pack(side=RIGHT)
    bE3.pack(side=RIGHT)
    bE4.pack(side=RIGHT)
    bE5.pack(side=RIGHT)
    bE6.pack(side=RIGHT)
    bE7.pack(side=RIGHT)
    bE8.pack(side=RIGHT)
    bE9.pack(side=RIGHT)
    bE10.pack(side=RIGHT)
    bE11.pack(side=RIGHT)
    bE12.pack(side=RIGHT)
    bE13.pack(side=RIGHT)
    bE14.pack(side=RIGHT)

    encStepText1.pack(side=RIGHT)
    encStepText2.pack(side=RIGHT)
    encStepText3.pack(side=RIGHT)
    encStepText4.pack(side=RIGHT)
    encStepText5.pack(side=RIGHT)
    encStepText6.pack(side=RIGHT)
    encStepText7.pack(side=RIGHT)
    encStepText8.pack(side=RIGHT)
    encStepText9.pack(side=RIGHT)
    encStepText10.pack(side=RIGHT)
    encStepText11.pack(side=RIGHT)
    encStepText12.pack(side=RIGHT)
    encStepText13.pack(side=RIGHT)
    encStepText14.pack(side=RIGHT)

    butttonToNext4.pack(side=RIGHT)
    buttonToExit4.pack(side=RIGHT)

    bigEncLabel1.pack(side=TOP)
    bigEncLabel2.pack(side=TOP)
    bigEncLabel3.pack(side=TOP)
    bigEncLabel4.pack(side=TOP)
    bigEncLabel5.pack(side=TOP)
    bigEncLabel6.pack(side=TOP)
    bigEncLabel7.pack(side=TOP)
    bigEncLabel8.pack(side=TOP)
    bigEncLabel9.pack(side=TOP)
    bigEncLabel10.pack(side=TOP)
    bigEncLabel11.pack(side=TOP)
```

```
    bigEncLabel12.pack(side=TOP)
    bigEncLabel13.pack(side=TOP)
    bigEncLabel14.pack(side=TOP)

    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 5
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 5
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 5
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 5
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 5
SDES ENCRYPTION
    # ------------------------------------------------------------------
------------------------------------------------------------PAGE 5
SDES ENCRYPTION


    topTitle4 = Label(sdesFrame3, borderwidth=0, highlightthickness=0,
text="S-DES Encryption\n", bg="salmon1",font=("Calibri", 24,"bold"))


    bigEnc2Label1 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label2 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label3 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label4 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label5 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label6 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label7 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label8 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label9 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label10 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label11 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigEnc2Label12 = Label(sdesFrame3, borderwidth=0,
highlightthickness=0,bg="salmon1")


    Enc2Step0  = Label(sdesFrame3, borderwidth=0,
highlightthickness=0, text="          (IP^-1)IP8-
inversed=[3,0,2,4,6,1,7,5], Permute4=[1,3,2,0], Expand And
Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of
the bits sequence. \n\nLoop 2 for key 2",bg="salmon1",
font=("Calibri", 17))
    Enc2Step1  = Label(bigEnc2Label1, borderwidth=0,
```

```
highlightthickness=0, text="Step 1:8-bit input after SW(swap):",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step2  = Label(bigEnc2Label2, borderwidth=0,
highlightthickness=0, text="Step 2:Divide to 2 halfs:", bg="salmon1",
font=("Calibri", 17),height=2,width=50)
    Enc2Step3  = Label(bigEnc2Label3, borderwidth=0,
highlightthickness=0, text="Step 3::Expand And Permutate right half:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step4  = Label(bigEnc2Label4, borderwidth=0,
highlightthickness=0, text="Step 4:Previous output XOR key 2:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    Enc2Step5  = Label(bigEnc2Label5, borderwidth=0,
highlightthickness=0, text="Step 5:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step6  = Label(bigEnc2Label6, borderwidth=0,
highlightthickness=0, text="Step 6:Left S0-box:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step7  = Label(bigEnc2Label7, borderwidth=0,
highlightthickness=0, text="Step 7:Right S1-box:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step8  = Label(bigEnc2Label8, borderwidth=0,
highlightthickness=0, text="Step 8:Combine the two new left and right
2-bits:    ", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step9  = Label(bigEnc2Label9, borderwidth=0,
highlightthickness=0, text="Step 9:Permute 4 last output:    ",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step10 = Label(bigEnc2Label10, borderwidth=0,
highlightthickness=0, text="Step 10::Last Output XOR Left half of
initial input:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step11 = Label(bigEnc2Label11, borderwidth=0,
highlightthickness=0, text="Step 11:Combine last output  right half of
initial input:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Enc2Step12 = Label(bigEnc2Label12, borderwidth=0,
highlightthickness=0, text="Step 12:PermuteIP^-1 8-bit-> Cipher
text:", bg="salmon1",font=("Calibri", 17),height=2,width=50)


    var2E1 = StringVar()
    var2E1.set("")
    var2E2 = StringVar()
    var2E2.set("")
    var2E3 = StringVar()
    var2E3.set("")
    var2E4 = StringVar()
    var2E4.set("")
    var2E5 = StringVar()
    var2E5.set("")
    var2E6 = StringVar()
    var2E6.set("")
    var2E7 = StringVar()
    var2E7.set("")
    var2E8 = StringVar()
    var2E8.set("")
    var2E9 = StringVar()
    var2E9.set("")
    var2E10 = StringVar()
    var2E10.set("")
    var2E11 = StringVar()
    var2E11.set("")
    var2E12 = StringVar()
    var2E12.set("")
```

```
    var2E13 = StringVar()
    var2E13.set("")
    var2E14 = StringVar()
    var2E14.set("")

    def call_E2s1():
        var2E1.set("8-bit input: "+''.join(encryptionsteps2[0]))

    def call_E2s2():
        var2E2.set("Left Half: " + ''.join(encryptionsteps2[1])+" ,
Right Half: "+''.join(encryptionsteps2[2]))

    def call_E2s3():
        var2E3.set("EP right half: " + ''.join(encryptionsteps2[3]))

    def call_E2s4():
        var2E4.set("EP XOR KEY 1: "+''.join(encryptionsteps2[4]))

    def call_E2s5():
        var2E5.set("Left Half: " + ''.join(encryptionsteps2[5])+" ,
Right Half: "+''.join(encryptionsteps2[6]))

    def call_E2s6():
        var2E6.set("LEFT S0 result: "+''.join(encryptionsteps2[7]))

    def call_E2s7():
        var2E7.set("Right S1 result: "+''.join(encryptionsteps2[8]))

    def call_E2s8():
        var2E8.set("Combination: "+''.join(encryptionsteps2[9]))

    def call_E2s9():
        var2E9.set("P4 Permute: " + ''.join(encryptionsteps2[10]))

    def call_E2s10():
        var2E10.set("P4 XOR  IP LEFT HALF: " +
''.join(encryptionsteps2[11]))

    def call_E2s11():
        var2E11.set("Combination of previous output and ip right half:
"+ ''.join(encryptionsteps2[12]))

    def call_E2s12():
        var2E12.set("Cipher result: "+ ''.join(encryptionsteps2[13]))



    enc2StepText1 = Label(bigEnc2Label1, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E1)
    enc2StepText2 = Label(bigEnc2Label2, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E2)
    enc2StepText3 = Label(bigEnc2Label3, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E3)
    enc2StepText4 = Label(bigEnc2Label4, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E4)
    enc2StepText5 = Label(bigEnc2Label5, bg="white", borderwidth=0,
```

```
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E5)
    enc2StepText6 = Label(bigEnc2Label6, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E6)
    enc2StepText7 = Label(bigEnc2Label7, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E7)
    enc2StepText8 = Label(bigEnc2Label8, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E8)
    enc2StepText9 = Label(bigEnc2Label9, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E9)
    enc2StepText10 = Label(bigEnc2Label10, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E10)
    enc2StepText11 = Label(bigEnc2Label11, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E11)
    enc2StepText12 = Label(bigEnc2Label12, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2E12)




    butttonToNext5=Button(sdesFrame3,text="Next",command=lambda:
raise_frame(sdesFrame4),bg="SteelBlue3",height = 3, width = 8)

buttonToExit5=Button(sdesFrame3,text="Back",command=lambda:raise_frame
(sdesFrame2),bg="SteelBlue3",height = 3, width = 8)
    b2E1 = Button(bigEnc2Label1, text="Step
1",command=lambda:call_E2s1(),bg="SteelBlue3",height = 1, width = 6)
    b2E2 = Button(bigEnc2Label2, text="Step
2",command=lambda:call_E2s2(),bg="SteelBlue3",height = 1, width = 6)
    b2E3 = Button(bigEnc2Label3, text="Step
3",command=lambda:call_E2s3(),bg="SteelBlue3",height = 1, width = 6)
    b2E4 = Button(bigEnc2Label4, text="Step
4",command=lambda:call_E2s4(),bg="SteelBlue3",height = 1, width = 6)
    b2E5 = Button(bigEnc2Label5, text="Step
5",command=lambda:call_E2s5(),bg="SteelBlue3",height = 1, width = 6)
    b2E6 = Button(bigEnc2Label6,text="Step
6",command=lambda:call_E2s6(),bg="SteelBlue3",height = 1, width = 6)
    b2E7 = Button(bigEnc2Label7, text="Step
7",command=lambda:call_E2s7(),bg="SteelBlue3",height = 1, width = 6)
    b2E8 = Button(bigEnc2Label8, text="Step
8",command=lambda:call_E2s8(),bg="SteelBlue3",height = 1, width = 6)
    b2E9 = Button(bigEnc2Label9,text="Step
9",command=lambda:call_E2s9(),bg="SteelBlue3",height = 1, width = 6)
    b2E10 = Button(bigEnc2Label10, text="Step
10",command=lambda:call_E2s10(),bg="SteelBlue3",height = 1, width = 6)
    b2E11 = Button(bigEnc2Label11, text="Step
11",command=lambda:call_E2s11(),bg="SteelBlue3",height = 1, width = 6)
    b2E12= Button(bigEnc2Label12, text="Step
12",command=lambda:call_E2s12(),bg="SteelBlue3",height = 1, width = 6)


    topTitle4.pack(side=TOP)
    Enc2Step0.pack(side=TOP)
```

```
    Enc2Step1.pack(side=LEFT)
    Enc2Step2.pack(side=LEFT)
    Enc2Step3.pack(side=LEFT)
    Enc2Step4.pack(side=LEFT)
    Enc2Step5.pack(side=LEFT)
    Enc2Step6.pack(side=LEFT)
    Enc2Step7.pack(side=LEFT)
    Enc2Step8.pack(side=LEFT)
    Enc2Step9.pack(side=LEFT)
    Enc2Step10.pack(side=LEFT)
    Enc2Step11.pack(side=LEFT)
    Enc2Step12.pack(side=LEFT)


    b2E1.pack(side=RIGHT)
    b2E2.pack(side=RIGHT)
    b2E3.pack(side=RIGHT)
    b2E4.pack(side=RIGHT)
    b2E5.pack(side=RIGHT)
    b2E6.pack(side=RIGHT)
    b2E7.pack(side=RIGHT)
    b2E8.pack(side=RIGHT)
    b2E9.pack(side=RIGHT)
    b2E10.pack(side=RIGHT)
    b2E11.pack(side=RIGHT)
    b2E12.pack(side=RIGHT)


    enc2StepText1.pack(side=RIGHT)
    enc2StepText2.pack(side=RIGHT)
    enc2StepText3.pack(side=RIGHT)
    enc2StepText4.pack(side=RIGHT)
    enc2StepText5.pack(side=RIGHT)
    enc2StepText6.pack(side=RIGHT)
    enc2StepText7.pack(side=RIGHT)
    enc2StepText8.pack(side=RIGHT)
    enc2StepText9.pack(side=RIGHT)
    enc2StepText10.pack(side=RIGHT)
    enc2StepText11.pack(side=RIGHT)
    enc2StepText12.pack(side=RIGHT)


    butttonToNext5.pack(side=RIGHT)
    buttonToExit5.pack(side=RIGHT)

    bigEnc2Label1.pack(side=TOP)
    bigEnc2Label2.pack(side=TOP)
    bigEnc2Label3.pack(side=TOP)
    bigEnc2Label4.pack(side=TOP)
    bigEnc2Label5.pack(side=TOP)
    bigEnc2Label6.pack(side=TOP)
    bigEnc2Label7.pack(side=TOP)
    bigEnc2Label8.pack(side=TOP)
    bigEnc2Label9.pack(side=TOP)
    bigEnc2Label10.pack(side=TOP)
    bigEnc2Label11.pack(side=TOP)
    bigEnc2Label12.pack(side=TOP)

##########################################################################
##########################################################################
################PAGE 6
```

```
################################################################################
################################################################################
################
################################################################################
################################################################################
################
################################################################################
################################################################################
################

    topTitle5 = Label(sdesFrame4, borderwidth=0, highlightthickness=0,
text="S-DES Decryption\n", bg="salmon1",font=("Calibri", 24,"bold"))


    bigDecLabel1 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel2 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel3 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel4 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel5 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel6 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel7 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel8 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel9 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel10 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel11 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel12 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0,bg="salmon1")
    bigDecLabel13 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDecLabel14 = Label(sdesFrame4, borderwidth=0,
highlightthickness=0, bg="salmon1")

    DecStep0  = Label(sdesFrame4, borderwidth=0, highlightthickness=0,
text="          Permute8=[5,2,6,3,7,4,9,8], Permute4=[1,3,2,0], Expand
And Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position
of the bits sequence. \n",bg="salmon1", font=("Calibri", 17))
    DecStep1  = Label(bigDecLabel1, borderwidth=0,
highlightthickness=0, text="Step 1:Cipher 8-bit input((IP^-1) of
encryption exit):", bg="salmon1",font=("Calibri",
17),height=2,width=50)
    DecStep2  = Label(bigDecLabel2, borderwidth=0,
highlightthickness=0, text="Step 2:Permute the Cipher 8-bit input:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    DecStep3  = Label(bigDecLabel3, borderwidth=0,
highlightthickness=0, text="Step 3:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep4  = Label(bigDecLabel4, borderwidth=0,
highlightthickness=0, text="Step 4:Expand And Permutate right half:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    DecStep5  = Label(bigDecLabel5, borderwidth=0,
```

```
highlightthickness=0, text="Step 5:Previous output XOR key 2:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep6  = Label(bigDecLabel6, borderwidth=0,
highlightthickness=0, text="Step 6:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep7  = Label(bigDecLabel7, borderwidth=0,
highlightthickness=0, text="Step 7:Left S0-box:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep8  = Label(bigDecLabel8, borderwidth=0,
highlightthickness=0, text="Step 8:Right S1-box:    ",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep9  = Label(bigDecLabel9, borderwidth=0,
highlightthickness=0, text="Step 9:Combine the two new left and right
2-bits:    ", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep10 = Label(bigDecLabel10, borderwidth=0,
highlightthickness=0, text="Step 10:Permute 4 last output:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep11 = Label(bigDecLabel11, borderwidth=0,
highlightthickness=0, text="Step 11:Last Output XOR Left half of
initial cipher input:", bg="salmon1",font=("Calibri",
17),height=2,width=50)
    DecStep12 = Label(bigDecLabel12, borderwidth=0,
highlightthickness=0, text="Step 12:Combine last output  right half of
initial cipher input:", bg="salmon1",font=("Calibri",
17),height=2,width=50)
    DecStep13 = Label(bigDecLabel13, borderwidth=0,
highlightthickness=0, text="Step 13:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    DecStep14 = Label(bigDecLabel14, borderwidth=0,
highlightthickness=0, text="Step 14:Swap Left & Right half's
Positions:", bg="salmon1",font=("Calibri", 17),height=2,width=50)

    varDe1 = StringVar()
    varDe1.set("")
    varDe2 = StringVar()
    varDe2.set("")
    varDe3 = StringVar()
    varDe3.set("")
    varDe4 = StringVar()
    varDe4.set("")
    varDe5 = StringVar()
    varDe5.set("")
    varDe6 = StringVar()
    varDe6.set("")
    varDe7 = StringVar()
    varDe7.set("")
    varDe8 = StringVar()
    varDe8.set("")
    varDe9 = StringVar()
    varDe9.set("")
    varDe10 = StringVar()
    varDe10.set("")
    varDe11 = StringVar()
    varDe11.set("")
    varDe12 = StringVar()
    varDe12.set("")
    varDe13 = StringVar()
    varDe13.set("")
    varDe14 = StringVar()
    varDe14.set("")
```

```
    def call_Des1():
        varDe1.set("8-bit input: "+''.join(decryptionsteps[0]))

    def call_Des2():
        varDe2.set("IP8 Permute: " + ''.join(decryptionsteps[1]))

    def call_Des3():
        varDe3.set("Left Half: " + ''.join(decryptionsteps[2])+" ,
Right Half: "+''.join(decryptionsteps[3]))

    def call_Des4():
        varDe4.set("EP right half: " + ''.join(decryptionsteps[4]))

    def call_Des5():
        varDe5.set("EP XOR KEY 1: "+''.join(decryptionsteps[5]))

    def call_Des6():
        varDe6.set("Left Half: " + ''.join(decryptionsteps[6])+" ,
Right Half: "+''.join(decryptionsteps[7]))

    def call_Des7():
        varDe7.set("LEFT S0 result: "+''.join(decryptionsteps[8]))

    def call_Des8():
        varDe8.set("Right S1 result: "+''.join(decryptionsteps[9]))

    def call_Des9():
        varDe9.set("Combination: "+''.join(decryptionsteps[10]))

    def call_Des10():
        varDe10.set("P4 Permute: " + ''.join(decryptionsteps[11]))

    def call_Des11():
        varDe11.set("P4 XOR  IP LEFT HALF: " +
''.join(decryptionsteps[12]))

    def call_Des12():
        varDe12.set("Combination of previous output and ip right half:
"+ ''.join(decryptionsteps[13]))

    def call_Des13():
        varDe13.set("Left Half: " + ''.join(decryptionsteps[14])+" ,
Right Half: "+''.join(decryptionsteps[15]))

    def call_Des14():
        varDe14.set("Swap L-R half's: "+''.join(decryptionsteps[16]))



    DecStepText1 = Label(bigDecLabel1, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe1)
    DecStepText2 = Label(bigDecLabel2, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe2)
    DecStepText3 = Label(bigDecLabel3, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe3)
    DecStepText4 = Label(bigDecLabel4, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe4)
```

```
    DecStepText5 = Label(bigDecLabel5, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe5)
    DecStepText6 = Label(bigDecLabel6, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe6)
    DecStepText7 = Label(bigDecLabel7, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe7)
    DecStepText8 = Label(bigDecLabel8, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe8)
    DecStepText9 = Label(bigDecLabel9, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe9)
    DecStepText10 = Label(bigDecLabel10, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe10)
    DecStepText11 = Label(bigDecLabel11, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe11)
    DecStepText12 = Label(bigDecLabel12, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe12)
    DecStepText13 = Label(bigDecLabel13, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe13)
    DecStepText14 = Label(bigDecLabel14, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=varDe14)



    butttonToNext6=Button(sdesFrame4,text="Next",command=lambda:
raise_frame(sdesFrame5),bg="SteelBlue3",height = 3, width = 8)

buttonToExit6=Button(sdesFrame4,text="Back",command=lambda:raise_frame
(sdesFrame3),bg="SteelBlue3",height = 3, width = 8)
    bDe1 = Button(bigDecLabel1, text="Step
1",command=lambda:call_Des1(),bg="SteelBlue3",height = 1, width = 6)
    bDe2 = Button(bigDecLabel2, text="Step
2",command=lambda:call_Des2(),bg="SteelBlue3",height = 1, width = 6)
    bDe3 = Button(bigDecLabel3, text="Step
3",command=lambda:call_Des3(),bg="SteelBlue3",height = 1, width = 6)
    bDe4 = Button(bigDecLabel4, text="Step
4",command=lambda:call_Des4(),bg="SteelBlue3",height = 1, width = 6)
    bDe5 = Button(bigDecLabel5, text="Step
5",command=lambda:call_Des5(),bg="SteelBlue3",height = 1, width = 6)
    bDe6 = Button(bigDecLabel6,text="Step
6",command=lambda:call_Des6(),bg="SteelBlue3",height = 1, width = 6)
    bDe7 = Button(bigDecLabel7, text="Step
7",command=lambda:call_Des7(),bg="SteelBlue3",height = 1, width = 6)
    bDe8 = Button(bigDecLabel8, text="Step
8",command=lambda:call_Des8(),bg="SteelBlue3",height = 1, width = 6)
    bDe9 = Button(bigDecLabel9,text="Step
9",command=lambda:call_Des9(),bg="SteelBlue3",height = 1, width = 6)
    bDe10 = Button(bigDecLabel10, text="Step
10",command=lambda:call_Des10(),bg="SteelBlue3",height = 1, width = 6)
    bDe11 = Button(bigDecLabel11, text="Step
11",command=lambda:call_Des11(),bg="SteelBlue3",height = 1, width = 6)
    bDe12= Button(bigDecLabel12, text="Step
```

```
12",command=lambda:call_Des12(),bg="SteelBlue3",height = 1, width = 6)
    bDe13 = Button(bigDecLabel13, text="Step
13",command=lambda:call_Des13(),bg="SteelBlue3",height = 1, width = 6)
    bDe14 = Button(bigDecLabel14, text="Step
14",command=lambda:call_Des14(),bg="SteelBlue3",height = 1, width = 6)

    topTitle5.pack(side=TOP)
    DecStep0.pack(side=TOP)

    DecStep1.pack(side=LEFT)
    DecStep2.pack(side=LEFT)
    DecStep3.pack(side=LEFT)
    DecStep4.pack(side=LEFT)
    DecStep5.pack(side=LEFT)
    DecStep6.pack(side=LEFT)
    DecStep7.pack(side=LEFT)
    DecStep8.pack(side=LEFT)
    DecStep9.pack(side=LEFT)
    DecStep10.pack(side=LEFT)
    DecStep11.pack(side=LEFT)
    DecStep12.pack(side=LEFT)
    DecStep13.pack(side=LEFT)
    DecStep14.pack(side=LEFT)

    bDe1.pack(side=RIGHT)
    bDe2.pack(side=RIGHT)
    bDe3.pack(side=RIGHT)
    bDe4.pack(side=RIGHT)
    bDe5.pack(side=RIGHT)
    bDe6.pack(side=RIGHT)
    bDe7.pack(side=RIGHT)
    bDe8.pack(side=RIGHT)
    bDe9.pack(side=RIGHT)
    bDe10.pack(side=RIGHT)
    bDe11.pack(side=RIGHT)
    bDe12.pack(side=RIGHT)
    bDe13.pack(side=RIGHT)
    bDe14.pack(side=RIGHT)

    DecStepText1.pack(side=RIGHT)
    DecStepText2.pack(side=RIGHT)
    DecStepText3.pack(side=RIGHT)
    DecStepText4.pack(side=RIGHT)
    DecStepText5.pack(side=RIGHT)
    DecStepText6.pack(side=RIGHT)
    DecStepText7.pack(side=RIGHT)
    DecStepText8.pack(side=RIGHT)
    DecStepText9.pack(side=RIGHT)
    DecStepText10.pack(side=RIGHT)
    DecStepText11.pack(side=RIGHT)
    DecStepText12.pack(side=RIGHT)
    DecStepText13.pack(side=RIGHT)
    DecStepText14.pack(side=RIGHT)

    butttonToNext6.pack(side=RIGHT)
    buttonToExit6.pack(side=RIGHT)

    bigDecLabel1.pack(side=TOP)
    bigDecLabel2.pack(side=TOP)
    bigDecLabel3.pack(side=TOP)
    bigDecLabel4.pack(side=TOP)
```

```
    bigDecLabel5.pack(side=TOP)
    bigDecLabel6.pack(side=TOP)
    bigDecLabel7.pack(side=TOP)
    bigDecLabel8.pack(side=TOP)
    bigDecLabel9.pack(side=TOP)
    bigDecLabel10.pack(side=TOP)
    bigDecLabel11.pack(side=TOP)
    bigDecLabel12.pack(side=TOP)
    bigDecLabel13.pack(side=TOP)
    bigDecLabel14.pack(side=TOP)

    # ---------------------------------------------------------------------
----------------------------------------------------------PAGE 7
SDES ENCRYPTION
    # ---------------------------------------------------------------------
----------------------------------------------------------PAGE 7
SDES ENCRYPTION
    # ---------------------------------------------------------------------
----------------------------------------------------------PAGE 7
SDES ENCRYPTION
    # ---------------------------------------------------------------------
----------------------------------------------------------PAGE 7
SDES ENCRYPTION
    # ---------------------------------------------------------------------
----------------------------------------------------------PAGE 7
SDES ENCRYPTION
    # ---------------------------------------------------------------------
----------------------------------------------------------PAGE 7
SDES ENCRYPTION


    topTitle6 = Label(sdesFrame5, borderwidth=0, highlightthickness=0,
text="S-DES Decryption\n", bg="salmon1",font=("Calibri", 24,"bold"))


    bigDec2Label1 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label2 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label3 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label4 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label5 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label6 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label7 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label8 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label9 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label10 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label11 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, bg="salmon1")
    bigDec2Label12 = Label(sdesFrame5, borderwidth=0,
highlightthickness=0,bg="salmon1")
```

```
    Dec2Step0  = Label(sdesFrame5, borderwidth=0,
highlightthickness=0, text="          (IP^-1)IP8-
inversed=[3,0,2,4,6,1,7,5], Permute4=[1,3,2,0], Expand And
Permutate=[3,0,1,2,1,2,3,0] where 0 to 1 is the element position of
the bits sequence. \n\nLoop 2 for key 1",bg="salmon1",
font=("Calibri", 17))
    Dec2Step1  = Label(bigDec2Label1, borderwidth=0,
highlightthickness=0, text="Step 1:8-bit input after SW(swap):",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step2  = Label(bigDec2Label2, borderwidth=0,
highlightthickness=0, text="Step 2:Divide to 2 halfs:", bg="salmon1",
font=("Calibri", 17),height=2,width=50)
    Dec2Step3  = Label(bigDec2Label3, borderwidth=0,
highlightthickness=0, text="Step 3::Expand And Permutate right half:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step4  = Label(bigDec2Label4, borderwidth=0,
highlightthickness=0, text="Step 4:Previous output XOR key 2:",
bg="salmon1", font=("Calibri", 17),height=2,width=50)
    Dec2Step5  = Label(bigDec2Label5, borderwidth=0,
highlightthickness=0, text="Step 5:Divide to 2 halfs:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step6  = Label(bigDec2Label6, borderwidth=0,
highlightthickness=0, text="Step 6:Left S0-box:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step7  = Label(bigDec2Label7, borderwidth=0,
highlightthickness=0, text="Step 7:Right S1-box:",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step8  = Label(bigDec2Label8, borderwidth=0,
highlightthickness=0, text="Step 8:Combine the two new left and right
2-bits:    ", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step9  = Label(bigDec2Label9, borderwidth=0,
highlightthickness=0, text="Step 9:Permute 4 last output:    ",
bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step10 = Label(bigDec2Label10, borderwidth=0,
highlightthickness=0, text="Step 10::Last Output XOR Left half of
initial input:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step11 = Label(bigDec2Label11, borderwidth=0,
highlightthickness=0, text="Step 11:Combine last output  right half of
initial input:", bg="salmon1",font=("Calibri", 17),height=2,width=50)
    Dec2Step12 = Label(bigDec2Label12, borderwidth=0,
highlightthickness=0, text="Step 12: PermuteIP^-1 8-bit-> -> De-
Ciphered text/bits:", bg="salmon1",font=("Calibri",
17),height=2,width=50)


    var2De1 = StringVar()
    var2De1.set("")
    var2De2 = StringVar()
    var2De2.set("")
    var2De3 = StringVar()
    var2De3.set("")
    var2De4 = StringVar()
    var2De4.set("")
    var2De5 = StringVar()
    var2De5.set("")
    var2De6 = StringVar()
    var2De6.set("")
    var2De7 = StringVar()
    var2De7.set("")
    var2De8 = StringVar()
    var2De8.set("")
```

```
    var2De9 = StringVar()
    var2De9.set("")
    var2De10 = StringVar()
    var2De10.set("")
    var2De11 = StringVar()
    var2De11.set("")
    var2De12 = StringVar()
    var2De12.set("")
    var2De13 = StringVar()
    var2De13.set("")
    var2De14 = StringVar()
    var2De14.set("")

    def call_De2s1():
        var2De1.set("8-bit input: "+''.join(decryptionsteps2[0]))

    def call_De2s2():
        var2De2.set("Left Half: " + ''.join(decryptionsteps2[1])+" ,
Right Half: "+''.join(decryptionsteps2[2]))

    def call_De2s3():
        var2De3.set("EP right half: " + ''.join(decryptionsteps2[3]))

    def call_De2s4():
        var2De4.set("EP XOR KEY 1: "+''.join(decryptionsteps2[4]))

    def call_De2s5():
        var2De5.set("Left Half: " + ''.join(decryptionsteps2[5])+" ,
Right Half: "+''.join(decryptionsteps2[6]))

    def call_De2s6():
        var2De6.set("LEFT S0 result: "+''.join(decryptionsteps2[7]))

    def call_De2s7():
        var2De7.set("Right S1 result: "+''.join(decryptionsteps2[8]))

    def call_De2s8():
        var2De8.set("Combination: "+''.join(decryptionsteps2[9]))

    def call_De2s9():
        var2De9.set("P4 Permute: " + ''.join(decryptionsteps2[10]))

    def call_De2s10():
        var2De10.set("P4 XOR  IP LEFT HALF: " +
''.join(decryptionsteps2[11]))

    def call_De2s11():
        var2De11.set("Combination of previous output and ip right
half: "+ ''.join(decryptionsteps2[12]))

    def call_De2s12():
        var2De12.set("Cipher result: "+ ''.join(decryptionsteps2[13]))



    Dec2StepText1 = Label(bigDec2Label1, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De1)
    Dec2StepText2 = Label(bigDec2Label2, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
```

```
16),height=1,textvariable=var2De2)
    Dec2StepText3 = Label(bigDec2Label3, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De3)
    Dec2StepText4 = Label(bigDec2Label4, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De4)
    Dec2StepText5 = Label(bigDec2Label5, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De5)
    Dec2StepText6 = Label(bigDec2Label6, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De6)
    Dec2StepText7 = Label(bigDec2Label7, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De7)
    Dec2StepText8 = Label(bigDec2Label8, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De8)
    Dec2StepText9 = Label(bigDec2Label9, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De9)
    Dec2StepText10 = Label(bigDec2Label10, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De10)
    Dec2StepText11 = Label(bigDec2Label11, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De11)
    Dec2StepText12 = Label(bigDec2Label12, bg="white", borderwidth=0,
highlightthickness=0,font=("Calibri",
16),height=1,textvariable=var2De12)




    butttonToNext7=Button(sdesFrame5,text="Next",command=lambda:
raise_frame(sdesFrame6),bg="SteelBlue3",height = 3, width = 8)

buttonToExit7=Button(sdesFrame5,text="Back",command=lambda:raise_frame
(sdesFrame4),bg="SteelBlue3",height = 3, width = 8)
    b2De1 = Button(bigDec2Label1, text="Step
1",command=lambda:call_De2s1(),bg="SteelBlue3",height = 1, width = 6)
    b2De2 = Button(bigDec2Label2, text="Step
2",command=lambda:call_De2s2(),bg="SteelBlue3",height = 1, width = 6)
    b2De3 = Button(bigDec2Label3, text="Step
3",command=lambda:call_De2s3(),bg="SteelBlue3",height = 1, width = 6)
    b2De4 = Button(bigDec2Label4, text="Step
4",command=lambda:call_De2s4(),bg="SteelBlue3",height = 1, width = 6)
    b2De5 = Button(bigDec2Label5, text="Step
5",command=lambda:call_De2s5(),bg="SteelBlue3",height = 1, width = 6)
    b2De6 = Button(bigDec2Label6,text="Step
6",command=lambda:call_De2s6(),bg="SteelBlue3",height = 1, width = 6)
    b2De7 = Button(bigDec2Label7, text="Step
7",command=lambda:call_De2s7(),bg="SteelBlue3",height = 1, width = 6)
    b2De8 = Button(bigDec2Label8, text="Step
8",command=lambda:call_De2s8(),bg="SteelBlue3",height = 1, width = 6)
    b2De9 = Button(bigDec2Label9,text="Step
9",command=lambda:call_De2s9(),bg="SteelBlue3",height = 1, width = 6)
    b2De10 = Button(bigDec2Label10, text="Step
10",command=lambda:call_De2s10(),bg="SteelBlue3",height = 1, width =
6)
```

```
    b2De11 = Button(bigDec2Label11, text="Step
11",command=lambda:call_De2s11(),bg="SteelBlue3",height = 1, width =
6)
    b2De12= Button(bigDec2Label12, text="Step
12",command=lambda:call_De2s12(),bg="SteelBlue3",height = 1, width =
6)


    topTitle6.pack(side=TOP)
    Dec2Step0.pack(side=TOP)

    Dec2Step1.pack(side=LEFT)
    Dec2Step2.pack(side=LEFT)
    Dec2Step3.pack(side=LEFT)
    Dec2Step4.pack(side=LEFT)
    Dec2Step5.pack(side=LEFT)
    Dec2Step6.pack(side=LEFT)
    Dec2Step7.pack(side=LEFT)
    Dec2Step8.pack(side=LEFT)
    Dec2Step9.pack(side=LEFT)
    Dec2Step10.pack(side=LEFT)
    Dec2Step11.pack(side=LEFT)
    Dec2Step12.pack(side=LEFT)


    b2De1.pack(side=RIGHT)
    b2De2.pack(side=RIGHT)
    b2De3.pack(side=RIGHT)
    b2De4.pack(side=RIGHT)
    b2De5.pack(side=RIGHT)
    b2De6.pack(side=RIGHT)
    b2De7.pack(side=RIGHT)
    b2De8.pack(side=RIGHT)
    b2De9.pack(side=RIGHT)
    b2De10.pack(side=RIGHT)
    b2De11.pack(side=RIGHT)
    b2De12.pack(side=RIGHT)


    Dec2StepText1.pack(side=RIGHT)
    Dec2StepText2.pack(side=RIGHT)
    Dec2StepText3.pack(side=RIGHT)
    Dec2StepText4.pack(side=RIGHT)
    Dec2StepText5.pack(side=RIGHT)
    Dec2StepText6.pack(side=RIGHT)
    Dec2StepText7.pack(side=RIGHT)
    Dec2StepText8.pack(side=RIGHT)
    Dec2StepText9.pack(side=RIGHT)
    Dec2StepText10.pack(side=RIGHT)
    Dec2StepText11.pack(side=RIGHT)
    Dec2StepText12.pack(side=RIGHT)


    butttonToNext7.pack(side=RIGHT)
    buttonToExit7.pack(side=RIGHT)

    bigDec2Label1.pack(side=TOP)
    bigDec2Label2.pack(side=TOP)
    bigDec2Label3.pack(side=TOP)
    bigDec2Label4.pack(side=TOP)
    bigDec2Label5.pack(side=TOP)
```

```
    bigDec2Label6.pack(side=TOP)
    bigDec2Label7.pack(side=TOP)
    bigDec2Label8.pack(side=TOP)
    bigDec2Label9.pack(side=TOP)
    bigDec2Label10.pack(side=TOP)
    bigDec2Label11.pack(side=TOP)
    bigDec2Label12.pack(side=TOP)


#########################################################################
#########################################################################
###############END PAGE 8
#########################################################################
#########################################################################
################
#########################################################################
#########################################################################
################
#########################################################################
#########################################################################
################

    topTitle7 = Label(sdesFrame6, borderwidth=0, highlightthickness=0,
text="S-DES Encryption/Decryption Finished\n\n\n\n\n\n\n\n",
bg="salmon1",font=("Calibri", 26,"bold"))
    notes = Label(sdesFrame6, borderwidth=0, highlightthickness=0,
text="Please choose one of the following options Back or Exit to
continue or exit the program!!!\n       Note: You can check the
Documents, Encrypted Documents and Decrypted Documents folders for the
results of S-DES, at the directory:\n" + mydir + "\n\n",bg="salmon1",
font=("Calibri", 20, "bold"))
    EndCredits = Label(sdesFrame6, borderwidth=0,
highlightthickness=0, bg="salmon1")
    Credits = Label(EndCredits, borderwidth=0, highlightthickness=0,
text="ΒΑΖΑΙΟΣ ΣΤΥΛΙΑΝΟΣ Α.Μ.:1054284 CEID",bg="salmon1",
font=("Calibri", 17,"bold"), height=2, width=45)
    ButtonLabel = Label(sdesFrame6, borderwidth=0,
highlightthickness=0, bg="salmon1")

    butttonToExit7 = Button(ButtonLabel, text="Exit", command=lambda:
sys.exit(), bg="SteelBlue3", height=3,width=8)
    buttonToBack7 = Button(ButtonLabel, text="Back", command=lambda:
raise_frame(sdesFrame5), bg="SteelBlue3", height=3,width=8)

    topTitle7.pack(side=TOP)
    notes.pack(side=TOP)
    ButtonLabel.pack(side=TOP)
    EndCredits.pack(side=BOTTOM)
    Credits.pack(side=RIGHT)
    butttonToExit7.pack(side=RIGHT)
    buttonToBack7.pack(side=RIGHT)




    container.pack(side=TOP)
    canvas.pack(side="left", expand=True)
    scrollbarv.pack(side=RIGHT, fill="y")
    scrollbarh.pack(side=LEFT, fill="y")
    main_window.mainloop()#----------------------------Put always to
end of frames
```

```
    #main()




def permute(Bitkey, permSequence):
    permKey=[]
    global p10key
    global p8key
    global p4key
    for i in permSequence:#in each loop i contains a content of list
permSequence from 0-> (len(permSequence)-1)
        permKey += Bitkey[i]

    if (len(permKey) == 10):
        p10key=permKey
        print("10,",p10key)
    elif (len(permKey) == 8):
        p8key=permKey
        print("8,",p8key)
    elif(len(permKey) == 4):
        p4key = permKey
        print("4,",p4key)
    else:
        print("permutated key size=%d \n",len(permKey))

def divPermKeys(key):
    global leftKey
    global rightKey


    if len(leftKey): #initialise the left and right key if they have
already elements from before
        leftKey.clear()
    if len(rightKey):
        rightKey.clear()
    x=0
    mid=len(key)/2
    mid=int(mid)
    print("mid",mid)
    while(x<mid):
        #leftKey.insert(x,key[x])
        #rightKey.insert(x,key[mid+x])
        leftKey += key[x]
        rightKey += key[mid+x]

        x=x+1

    print("leftkey",leftKey)
    print("rightkey",rightKey)

def RoundShift(key):
    l=len(key)
    temp=key.copy()
    x=0
    print("l=",l)
    print("temp=",temp)
```

```
    while(x<l):
        #print("x=",x)
        if (x!=l-1):
            key[x]=temp[x+1]
        else:
            key[x]=temp[0]
        x=x+1
    print("shifted key=",key)
    return key

def XOR(list1,list2):
    xorRes=[]
    x=0
    while x<(len(list1)):
        if (list1[x]==list2[x]):
            #xorRes[x]=0
            xorRes.insert(x,"0")
        else:
            #xorRes[x] = 1
            xorRes.insert(x, "1")

        x=x+1
    #xorRes = tuple(xorRes)
    return xorRes

def BintoDec(binaryNum):
    x=0
    dec=0
    l=(len(binaryNum))
    print("binNumlen=",l)
    while(x<l):
        print("binNum ",x," =",binaryNum[x])
        dec=dec+(int(binaryNum[x])*(pow(2,(l-1-x))))
        x=x+1
    print("dec=",dec)
    return dec


def SboxMatrixChoise(fourBitKey,Sbox):
    print("4bitkey",fourBitKey)
    rowBinary=fourBitKey[0]+fourBitKey[3]
    print(rowBinary)
    columnBinary=fourBitKey[1]+fourBitKey[2]
    print(columnBinary)

    rowDec=(BintoDec(rowBinary))
    columnDec=(BintoDec(columnBinary))
    print(rowDec)
    print(columnDec)
    choise=Sbox[rowDec][columnDec]
    return choise

def correct_variable(value):
    temp1=[]
    y=0
    while (y < len(value)):
        temp1.insert(y, value[y])
        y=y+1
    correctValue=''.join(temp1)
    temp1.clear()
    return correctValue
```

```
def keyProduction():
    global key
    global leftKey
    global rightKey
    global combKey
    global combKey2
    global keysteps

    permute(key,p10Sequence) #permute the initial 10bit key
    keysteps[0]=correct_variable(p10key)
    divPermKeys(p10key) #divide key to 2 halfs
    keysteps[1]=correct_variable(leftKey)
    keysteps[2] =correct_variable(rightKey)
    leftKey=RoundShift(leftKey)#shift left half
    keysteps[3] = correct_variable(leftKey)
    rightKey=RoundShift(rightKey)#shift right half
    keysteps[4] = correct_variable(rightKey)
    combKey=leftKey+rightKey #combined key after shift for each half
    keysteps[5]=correct_variable(combKey)
    permute(combKey,p8Sequence) #permute the 8 bit key
    k1=p8key
    print("k1=",k1)
    ##############################################we now have our first
key
    leftKey=RoundShift(leftKey)#shift left half
    keysteps[6] = correct_variable(leftKey)
    leftKey=RoundShift(leftKey)#shift left half
    keysteps[7] = correct_variable(leftKey)
    rightKey=RoundShift(rightKey)#shift right half
    keysteps[8] = correct_variable(rightKey)
    rightKey=RoundShift(rightKey)#shift right half
    keysteps[9] = correct_variable(rightKey)
    combKey2=leftKey+rightKey
    keysteps[10] = correct_variable(combKey2)
    permute(combKey2,p8Sequence) #permute the 8 bit key
    k2=p8key
    print("k2=",k2)

    return (k1,k2)


def list_group_8(list):
    groups_num=len(list)/8
    group8=[]
    list_temp=[]
    w=0
    c=0
    while w<int(groups_num):
        list_temp=list[c:(c+8)]
        group8.insert(int(groups_num),list_temp)
        c=c+8
        w=w+1
    print("GROUP8=",group8)
    return group8
```

```
############################################################Encryption
Starts From Here




def Encryption_SDES(key,initPermBits):
    global count
    global iteration_count
    global dec_iteration_count
    global leftKey
    global rightKey
    if (count==0): #count=0 means 1st iteration cause in the second we
dont need permute in line180 and divPermKeys of p8Key but of last
value SW
        print("init permute IP")
        if(iteration_count == 1):
            encryptionsteps[0]=correct_variable(initPermBits)
        if(dec_iteration_count == 1):
            decryptionsteps[0] = correct_variable(initPermBits)

        permute(initPermBits,IP8)#p8key is palintext 8bit permuted
        if (iteration_count == 1):
            encryptionsteps[1]=correct_variable(p8key)
        if(dec_iteration_count == 1):
            decryptionsteps[1] = correct_variable(p8key)

        divPermKeys(p8key)
        if (iteration_count == 1):
            encryptionsteps[2] =correct_variable(leftKey)
            encryptionsteps[3] =correct_variable(rightKey)
        if(dec_iteration_count == 1):
            decryptionsteps[2] = correct_variable(leftKey)
            decryptionsteps[3] = correct_variable(rightKey)

    else:
        if (iteration_count == 1):
            encryptionsteps2[0] = correct_variable(initPermBits)
        if(dec_iteration_count == 1):
            decryptionsteps2[0] = correct_variable(initPermBits)

        divPermKeys(initPermBits)#-----------------------------------
-------------------------------------------------------------------
--
        if (iteration_count == 1):
            encryptionsteps2[1] = correct_variable(leftKey)
            encryptionsteps2[2] = correct_variable(rightKey)
        if(dec_iteration_count == 1):
            decryptionsteps2[1] = correct_variable(leftKey)
            decryptionsteps2[2] = correct_variable(rightKey)


    cin=0
    if(cin==0):
        y=0
        IPleft=[]
        IPright=[]
        while(y<len(leftKey)):
            IPleft.insert(y, leftKey[y])#gonna need it later initial
```

```
permutation left 4bit
            IPright.insert(y, rightKey[y])#gonna need it later initial
permutation right 4bit


            y=y+1
        cin=1


    #print("left1=", IPleft)
    #print()

    permute(rightKey,expandAndPermutate) #p8key has rightkey expanded
to 8 bit according to E.P. sequence
    if (count == 0 and iteration_count == 1):
        encryptionsteps[4] = correct_variable(p8key)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[4] = correct_variable(p8key)


    if(count == 1 and iteration_count == 1):
        encryptionsteps2[3] = correct_variable(p8key)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[3] = correct_variable(p8key)

    #print("left2=", IPleft)
    xor= XOR(p8key,key) #xorRes saved to xor variable
    if (count == 0 and iteration_count == 1):
        encryptionsteps[5] = correct_variable(xor)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[5] = correct_variable(xor)

    if(count == 1 and iteration_count == 1):
        encryptionsteps2[4] = correct_variable(xor)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[4] = correct_variable(xor)
    print("xor result=",xor)
    #print("left3=", IPleft)

    divPermKeys(xor) #re-values leftKey and rightKey
    if (count == 0 and iteration_count==1):
        encryptionsteps[6] = correct_variable(leftKey)
        encryptionsteps[7] = correct_variable(rightKey)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[6] = correct_variable(leftKey)
        decryptionsteps[7] = correct_variable(rightKey)

    if(count == 1 and iteration_count == 1):
        encryptionsteps2[5] = correct_variable(leftKey)
        encryptionsteps2[6] = correct_variable(rightKey)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[5] = correct_variable(leftKey)
        decryptionsteps2[6] = correct_variable(rightKey)

    #print("left4=", IPleft)
    twoBitLeft=SboxMatrixChoise(leftKey,S0) #leftKey to S0
    #print("left5=", IPleft)
    twoBitLeft=list(twoBitLeft)
    if (count == 0 and iteration_count == 1):
        encryptionsteps[8] = correct_variable(twoBitLeft)
    if(count == 0 and dec_iteration_count == 1):
```

```
        decryptionsteps[8] = correct_variable(twoBitLeft)


    if(count == 1 and iteration_count == 1):
        encryptionsteps2[7] = correct_variable(twoBitLeft)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[7] = correct_variable(twoBitLeft)
    #print("left6=", IPleft)

    twoBitRight=SboxMatrixChoise(rightKey,S1) #rightKey to S1
    twoBitRight=list(twoBitRight)
    if (count == 0 and iteration_count == 1):
        encryptionsteps[9] = correct_variable(twoBitRight)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[9] = correct_variable(twoBitRight)


    if(count==1 and iteration_count==1):
        encryptionsteps2[8] = correct_variable(twoBitRight)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[8] = correct_variable(twoBitRight)


    combTwoToFourBit=twoBitLeft+twoBitRight #combination of two 2bits
from Sboxes results left+right
    if (count == 0 and iteration_count==1):
        encryptionsteps[10] = correct_variable(combTwoToFourBit)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[10] = correct_variable(combTwoToFourBit)

    if(count==1 and iteration_count==1):
        encryptionsteps2[9] = correct_variable(combTwoToFourBit)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[9] = correct_variable(combTwoToFourBit)


    permute(combTwoToFourBit,p4Sequence) #p4key has thw 4bit permuted
value
    if (count == 0 and iteration_count==1):
        encryptionsteps[11] = correct_variable(p4key)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[11] = correct_variable(p4key)

    if(count==1 and iteration_count==1):
        encryptionsteps2[10] = correct_variable(p4key)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[10] = correct_variable(p4key)
    #print("left7=",IPleft)
    print("p4key=",p4key)

    xor=XOR(IPleft,p4key) #XOR of initial left 4bits with P4
    if (count == 0 and iteration_count == 1):
        encryptionsteps[12] = correct_variable(xor)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[12] = correct_variable(xor)

    if(count == 1 and iteration_count == 1):
        encryptionsteps2[11] = correct_variable(xor)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[11] = correct_variable(xor)
    print("ipleftpermXORp4key=",xor)
```

```
    combFourToEight=xor+IPright#step 10
    if (count == 0 and iteration_count == 1):
        encryptionsteps[13] = correct_variable(combFourToEight)
    if(count == 0 and dec_iteration_count == 1):
        decryptionsteps[13] = correct_variable(combFourToEight)


    if(count==1 and iteration_count == 1):
        encryptionsteps2[12] = correct_variable(combFourToEight)
    if(count == 1 and dec_iteration_count == 1):
        decryptionsteps2[12] = correct_variable(combFourToEight)


    if (count == 0):
        divPermKeys(combFourToEight) #step11
        if (iteration_count == 1):
            encryptionsteps[14] = correct_variable(leftKey)
            encryptionsteps[15] = correct_variable(rightKey)
        if(dec_iteration_count == 1):
            decryptionsteps[14] = correct_variable(leftKey)
            decryptionsteps[15] = correct_variable(rightKey)

        combFourToEight=rightKey+leftKey #step 12  bsk mporoume
apeu8eias---------> combFourToEight=IPright+xor
        if (iteration_count == 1):
            encryptionsteps[16] = correct_variable(combFourToEight)
        if(dec_iteration_count == 1):
            decryptionsteps[16] = correct_variable(combFourToEight)


    if (count == 0):
        finalBits=combFourToEight

    if (count==1):
        print("fin-1",combFourToEight)
        permute(combFourToEight, IP8reversed)
        print("itcount=",iteration_count)

        if (count == 1 and iteration_count == 1):
            print("mesa sto ip-1")
            encryptionsteps2[13] = correct_variable(p8key)

        if(count == 1 and dec_iteration_count == 1):
            decryptionsteps2[13] = correct_variable(p8key)

        print("itcount=", iteration_count)
        finalBits=p8key


    count = count + 1#increase count for loops count=0 first loop
count=2 second loop

    if len(IPleft):  # initialise the left and right key if they have
already elements from before
        IPleft.clear()
    if len(IPright):
        IPright.clear()


    return finalBits
```

```
#end OF ENCRYPTION\DECRYPTION ----------------------------------------
--------------------------------------------------------------------
-----------------


def main():
    global count
    global key
    global cipher_file_name
    global plaintext
    global key1
    global key2
    global iteration_count
    global dec_iteration_count
    global addedBits


    iteration_count=1
    dec_iteration_count = 1  # to start saving vars for buttons at ui
for first8 bit decryption
    bits_to_cipher=list_group_8(plaintext)# returns a list splited by
8 bits
    if not len(bits_to_cipher):
        unchecked_option()
    cipher_result = []
    Decrypt_cipher_result = []
    decrypted_final_without_added_bits=[]

    for i in bits_to_cipher:
        print("i=",i)
        plaintext=''.join(i)
        encryptKeys = keyProduction()
        key1 = tuple(encryptKeys[0])
        key2 = tuple(encryptKeys[1])

        #encryption-----------------------------------------------------
------------
        bitsIterationOne=Encryption_SDES(key1,plaintext)
        print("key1 exit SW=",bitsIterationOne)
        bitsIterationTwo=Encryption_SDES(key2,bitsIterationOne)
        count = 0
        iteration_count = iteration_count + 1
        cipherText=bitsIterationTwo
        cipher_result.append((''.join(cipherText)))
        #-------------------------------------------------------------
end encryption

        #decryption-----------------------------------------------------
----------------
        bitsDecryptIterationOne = Encryption_SDES(key2, cipherText)
        print("key1de exit SW=", bitsDecryptIterationOne)
        bitsDecryptIterationTwo = Encryption_SDES(key1,
bitsDecryptIterationOne)
        dec_iteration_count = dec_iteration_count + 1
        count = 0
        DecryptcipherText = bitsDecryptIterationTwo
        Decrypt_cipher_result.append((''.join(DecryptcipherText)))
        #-------------------------------------------------------end
decryption
```

```
        print("itercount=",iteration_count)
        print(cipherText)
        print(keysteps)

    if addedBits > 0:  # if we have added bits to be group of 8 remove
them for final result

edit_these_8_bits=Decrypt_cipher_result[(len(Decrypt_cipher_result)-
1)]
        n = 0
        while n < (len(edit_these_8_bits) - addedBits):

decrypted_final_without_added_bits.append(edit_these_8_bits[n])
            n = n + 1
        Decrypt_cipher_result.pop((len(Decrypt_cipher_result)-1))

Decrypt_cipher_result.append((''.join(decrypted_final_without_added_bi
ts))) #final decrypted value without added bits


    if len(cipher_file_name):#if file is input
        cipher_save_directory=(mydir + "\\Encrypted
Documents"+"\\""Encrypted_file_"+ cipher_file_name +".txt")
        cipher_file=open(cipher_save_directory,"w")
        cipher_file.write((''.join(cipher_result)))
        cipher_file.close()
        cipher_result.clear()

        decrypt_save_directory = (mydir + "\\Decrypted Documents" +
"\\" + "Decrypted_file_"+ cipher_file_name + ".txt")
        decrypt_file = open(decrypt_save_directory, "w")
        Decrypt_cipher_result_string =
''.join(Decrypt_cipher_result)#make bit sequense as one string
        print("decryted1", Decrypt_cipher_result)
        binTocharacher = binTostr(Decrypt_cipher_result_string)#bit
sequence to character
        decrypt_file.write(binTocharacher)
        decrypt_file.close()
        print(binTocharacher)
        Decrypt_cipher_result.clear()

    else:
        now = datetime.now()
        date_time = now.strftime("%d.%m.%Y, %H.%M.%S")
        if (kind == "text"):
            cipher_save_directory=(mydir + "\\Encrypted
Documents"+"\\"+"Encryption__text_string_"+ date_time +".txt")
            cipher_file=open(cipher_save_directory,"w")
            cipher_file.write((''.join(cipher_result)))
            cipher_file.close()
            print("cipher_result=",cipher_result)
            cipher_result.clear()

        if (kind == "binary"):
            cipher_save_directory=(mydir + "\\Encrypted
Documents"+"\\"+"Encryption_binary_"+ date_time +".txt")
            cipher_file=open(cipher_save_directory,"w")
            cipher_file.write((''.join(cipher_result)))
            cipher_file.close()
```

```
            cipher_result.clear()

        if(kind == "text"):
            decrypt_save_directory = (mydir + "\\Decrypted Documents"
+ "\\" + "Decryption_text_string_" + date_time + ".txt")
            decrypt_file = open(decrypt_save_directory, "w")
            Decrypt_cipher_result_string =
''.join(Decrypt_cipher_result)  # make bit sequense as one string
            print("decryted1", Decrypt_cipher_result)
            binTocharacher = binTostr(Decrypt_cipher_result_string)  #
bit sequence to character
            decrypt_file.write(binTocharacher)
            decrypt_file.close()
            print(binTocharacher)
            print("decrytedt", Decrypt_cipher_result)
            Decrypt_cipher_result.clear()

        if(kind == "binary"):
            decrypt_save_directory = (mydir + "\\Decrypted Documents"
+ "\\" + "Decryption_binary_" + date_time + ".txt")
            decrypt_file = open(decrypt_save_directory, "w")
            decrypt_file.write(('').join(Decrypt_cipher_result)))
            decrypt_file.close()
            print("decrytedb", Decrypt_cipher_result)
            Decrypt_cipher_result.clear()


main_Encrypt_window()
#ΒΑΖΑΙΟΣ ΣΤΥΛΙΑΝΟΣ Α.Μ.: 1054284
```