Aplikacja szyfrująca wykorzystująca szyfrowanie AES

# Register

Okno programu pozwalające zarejestrować użytkownika. Po jego uruchomieniu użytkownik zostanie poproszony o wybranie niezajętego nicku oraz wpisania dwu krotnie hasła zabezpieczającego do konta.

private bool ValidatePassword(string password)

Funkcja walidująca czy hasło wybrane przez użytkownika zawiera co najmniej 8 znaków, cyfrę, małą oraz dużą literę i znak specjalny.

string password hasło do zwalidowania

funkcja zwraca prawdę jeżeli hasło zawiera się w parametrach inaczej zwraca fałsz

private void Registerbutton\_Click(object sender, RoutedEventArgs e)

Funkcja uruchamiająca się po wciśnięciu przycisku Register lub enter

Najpierw, jeżeli nie istnieje ogólny plik użytkowników jest on utworzony inaczej pobierane są z niego nazwy wszystkich użytkowników. Plik UsersFiles\UserList.txt   
Funkcja następnie sprawdza czy nick jest wolny oraz czy wszystkie pola są wypełnione jeżeli tak wywołuję następną funkcje CreateUser.

private void CreateUser(string login,string password)

Zapisuje hasło/sól, użytkownika oraz klucze rsa do odpowiadających im folderów.

Wygenerowana zostaje 8 bitowa sól, która jest użyta do dodatkowego zahaschowania hasła. Zapisana jest w pliku User/salt.txt

Z hasła zahaszowanego solą utworzony jest skrót za pomocą algorytmu SHA2-512 i zapisany do pliku User/paswd.txt

Następnie utworzona jest para kluczy RSA 2048 bitów, które zostają zserializowane do xml a następnie zapisane

Klucz publiczny w User/PUGB/PUGB.txt

Klucz prywatny jest najpierw zaszyfrowany za pomocą algorytmu AES-ECB gdzie hasłem jest skrót hasła użytkownika a następnie zapisany w User/PRIV/PRIV.txt

SendCommand\_CanExecute

SendCommand\_Execute  
Funkcjonalność interfejsu wciśnij enter żeby zaakceptować.

# Login

Okno programu pozwalające się zalogować lub otworzyć okno rejestrowania.

private void Loginbutton\_Click(object sender, RoutedEventArgs e)

Funkcja uruchamiająca się po wciśnięciu login lub enter

Najpierw, jeżeli nie istnieje ogólny plik użytkowników jest on utworzony inaczej pobierane są z niego nazwy wszystkich użytkowników. Plik UsersFiles\UserList.txt   
Funkcja następnie sprawdza czy nick jest poprawny i istnieje oraz czy wszystkie pola są wypełnione.

Jeżeli tak to uruchamia się następna funkcja AuthenticateUser

private void AuthenticateUser(string login,string passwrd)

Funkcja najpierw pobiera sól z pliku użytkownika następnie haszhuje nią podane hasło funkcją skrótu SHA2-512 i porównuje skrót z wynikiem hasła zapisanym w pliku.

Jeżeli nastąpi poprawna autentykacja Okno zwróci login obecnie zalogowanego użytkownika oraz zamknie okno i otworzy główne okno programu.

# SHA2salted

Klasa wspomagająca zajmująca się tworzeniem skrótu SHA2-512 z podanego tekstu.

public static string GenerateSHA512String(string inputString, byte[] salt)

Funckja tworząca skrót z podanego tekstu w wykorzystaniem podanej soli. Zwraca hasło w postaci stringa.

public static string GetStringFromHash(byte[] hash)

Funkcja zamieniająca Hash na string zwraca string

# RSAHandle

Klasa wspomagająca zajmująca się ogólnym kodowaniem RSA.

public static void EncryptPrivate(string thingToEncrypt,string keyPassword,string whereToSave)

Funkcja szyfrująca AES-256 Rijndael ECB używająca podanego parametru, jako hasła szyfrowania i zapisująca wynik w nowo utworzonym pliku pod daną lokalizacją

Wykorzystywana głównie do enkrypcji klucza prywatnego RSA.

public static string DecryptPrivate( string keyPassword, string whereToRead)

Funkcja deszyfrująca AES-256 Rijndael ECB używająca podanego parametru, jako hasła deszyfrowania i zapisująca wynik w nowo utworzonym stringu którego zwraca

Wykorzystywana głównie do dekrypcji klucza prywatnego RSA.

public static RSAParameters StringToKey(string key)

Funkcja zamieniająca string na klucz RSA. Zwraca klucz RSA

public static string EncryptMessage(string publicKey, string toEncrypt)

Funkcja szyfrująca RSA podany tekst za pomocą podanego klucza, który najpierw jest zamieniony z postaci stringa do postaci klucza RSA. Zwraca zaszyfrowany tekst w postaci stringa.

public static string DecryptMessage(string privateKey, string toDecrypt)

Funkcja deszyfrująca RSA podany tekst za pomocą podanego klucza, który najpierw jest zamieniony z postaci stringa do postaci klucza RSA. Zwraca deszyfrowany tekst w postaci stringa.

# MainWindow

Okno pozwalające wybrać parametry szyfrowania, pliki i ich lokalizację oraz odbiorców. Przy uruchomieniu pobiera listę wszystkich użytkowników i wpisuje ich do książki adresowej.

private void fileSelectButton\_Click(object sender, RoutedEventArgs e)

Funkcja wyboru pliku do szyfrowania. Guzik wybierz plik.

private void pathSelectButton\_Click(object sender, RoutedEventArgs e)

Funkcja wyboru folderu do zapisu wyniku szyfrowania. Guzik wybierz gdzie zapisać.

private void addPersonButton\_Click(object sender, RoutedEventArgs e)

private void removePersonButton\_Click(object sender, RoutedEventArgs e)

Guziki strzałek. Służą do dodawania lub odejmowania ludzi do książki odbiorców.

private void magicButton\_Click(object sender, RoutedEventArgs e)

Guzik wykonania szyfrowania/deszyfracji

Najpierw sprawdza czy wszystkie wymagane parametry zostały zaznaczone a następnie zależnie od zaznaczenia szyfruj/deszyfruj wywołuje funkcję szyfrowania/deszyfrowania na podanym pliku, wynik zapiszę w podane miejsce z podaną nazwą, metoda szyfrowania zostanie wybrana z listy razem z długością klucza a następnie podana lista odbiorców.

# CipherMahine

Klasa zajmująca się ogólnym procesem szyfrowania i wszystkimi powiązanymi z tym problemami. Jak coś nie działa to najpewniej tutaj.

public void AES\_Encrypt(string originFile,string whereToSave,string logedUser,string mode, string keySize,List<string> selectedUsers)

Funkcja szyfrująca wymaga ścieżek do plików zalogowanego użytkownika trybu szyfrowania, rozmiaru klucza i adresatów.

Najpierw generuje ciąg bloku jawnego IV i hasło w postaci stringów używa do tego funkcji GetUniqueKey(), które są następnie zamieniane na bajty do maszyny szyfrującej.

IV ma 16 bajtów a hasło sesyjne 64

Następnie generowana jest 8 bajtowa sól za pomocą RNGCryptoServiceProvider

Z wszystkich podanych informacji generowany jest nagłówek jawny pliku szyfrowanego.

Zostaje obliczony rozmiar nagłówka w bajtach i wyliczona liczba czterocyfrowa(z wiodącymi zerami) zostaje wpisana na początek pliku.

Teraz wpisany zostaje nagłówek, jako tekst jawny a następnie wpisana zostaje sól.

Do głównego mechanizmu szyfrowania RijndaelManaged AES zostają teraz podane parametry.

Rozmiar klucza podany przez użytkownika 128/192/256 rozmiar bloku 128 padding PaddingMode.PKCS7 klucz zostaje wygenerowany za pomocą algorytmu Rfc2898 który wykonuje 10000 pętli, IV zostaje podany z poprzedniego kroku razem z solą oraz zostaje wybrany tryb szyfrowania CBC/ECB/CFB/OFB.

Na podstawie powyższego szyfratora zostaje utworzony strumień 1 mb, który odczytuje dane do szyfrowania i wpisuje drugim strumieniem 1mb do nowego pliku.

Strumienie zostają zamknięte.

public void AES\_Decrypt(string originFile, string whereToSave, string logedUser, string aPanDoKogo)

Funkcja deszyfrująca wymaga ścieżek do plików zalogowanego użytkownika i adresata.

Najpierw odczytywany jest 4 cyfrowy z wiodącymi zerami rozmiar jawnego nagłówka a następnie sam nagłówek.   
Nagłówek zostaje z parsowany i pobrane zostają z niego automatycznie parametry do odszyfrowania.

Szukane jest też czy zalogowana osoba znajduje się na liście odbiorców i czy wybrała siebie jako odbiorcę.

Jeżeli tak to klucz prywatny jest odczytywany z zapisanego miejsca i odszyfrowany za pomocą RSAHandle.DecryptPrivate. Uzyskany klucz jest wtedy użyty do odszyfrowania klucza sesyjnego za pomocą RSAHandle.DecryptMessage.

Z zaszyfrowanego pliku odczytujemy sól a następnie zamieniamy klucz sesyjny na bajty Rfc2898DeriveBytes i używamy go, jako klucza odszyfrowania następnie wybieramy pozostałe parametry.

Na podstawie powyższego deszyfratora zostaje utworzony strumień 1 mb, który odczytuje dane do odszyfrowania i wpisuje drugim strumieniem 1mb do nowego pliku, jako jawny tekst.

Strumienie zostają zamknięte

private string HeaderToString(string password, string mode, string keySize, List<string> selectedUsers,string IV)

Funkcja zajmująca się utworzeniem jawnego nagłówka do pliku szyfrowanego. Zwraca stringa z tekstem wynikowym.

Zamienia podane parametry szyfrowana na teks w podanej strukturze następnie dla każdego odbiorcy odnajduje jego klucz publiczny w postaci tekstu, zamienia go na klucz RSA i z jego pomocą szyfruje klucz sesyjny RSAHandle.EncryptMessage.

Zaszyfrowany klucz sesyjny jest wpisany do nagłówka.

public static string GetUniqueKey(int maxSize)

Funkcja pomocnicza tworząca unikatowy klucz o podanej ilości bajtów i zwracająca jego reprezentacje, jako string.

Używa RNGCryptoServiceProvider do generowania losowego tekstu.

# Źródła Pomocnicze

* Szyfracja AES na dużych plikach <https://stackoverflow.com/questions/27645527/aes-encryption-on-large-files>
* Szyfracja AES https://msdn.microsoft.com/pl-pl/library/system.security.cryptography.rijndaelmanaged(v=vs.110).aspx
* Enkrypcja RSA <https://stackoverflow.com/questions/17128038/c-sharp-rsa-encryption-decryption-with-transmission>
* <https://msdn.microsoft.com/pl-pl/library/system.security.cryptography.rsacryptoserviceprovider(v=vs.110).aspx>
* Password Hash Functions <https://en.wikipedia.org/wiki/PBKDF2>
* Poprawne praktyki Haszowania Haseł <https://crackstation.net/hashing-security.htm>