Dokumentacja aplikacji stworzonej na przedmiot Bezpieczeństwo Systemów Komputerowych

1. Wstęp

1.1. Autorzy oraz repozytorium

Patryk Milewski patryk.milewski@gmail.com
Stanisław Barański stasbar@gmail.com

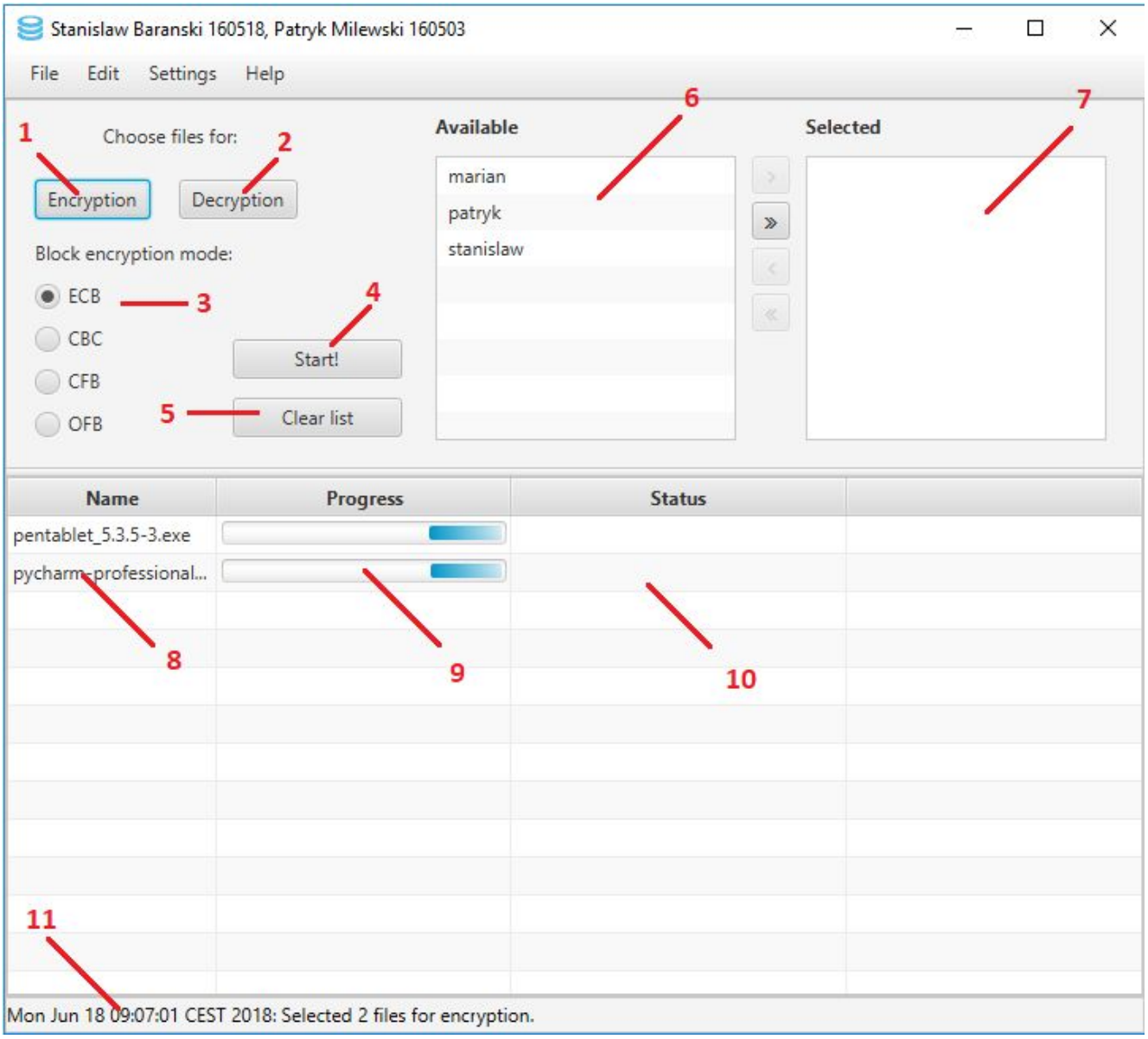
Repozytorium aplikacji:
https://github.com/stasbar/bsk

1.2. Opis aplikacji

Aplikacja ma na celu szyfrowanie i deszyfrowanie wybranych plików za pomocą symetrycznego algorytmu AES w różnych trybach blokowych: ECB, CBC, CFB i OFB. Została stworzna w całości w środowisku Java z użyciem frameworka JavaFX, który odpowiada za “okienkowość” aplikacji. Wymagania funkcjonalne zostały dostarczone zgodnie z wymaganiami do projektu z przedmiotu.

Zdecydowaliśmy się na wybór takiej formy aplikacji ze względu na hermetyczność i prostotę. Wszystko dzieje się lokalnie, bez udziału zewnętrznych usługodawców i komunikacji z nimi, co znacząco podnosi poziom bezpieczeństwa i zmniejsza skomplikowanie całości.

1.3. Opis interfejsu głównego



Po włączeniu aplikacji powinniśmy się najpierw zalogować się, co pozwoli nam na skorzystanie z pozostałych funkcji aplikacji. Możemy tego dokonać korzystając z gónego menu i wybierając “File”. W tym samym miejscu również znajduje się opcja wylogowania. Opis pozostałych elementów interfejsu:

1. Wybór plików do szyfrowania.
2. Wybór plików do odszyfrowania.
3. Wybór rodzaju trybu blokowego AES.
4. Rozpoczęcie szyfrowania i odszyfrowywania wybranych wcześniej plików.
5. Wyczyszczenie listy wybranych plików.
6. Lista użytkowników możliwych do wybrania, którym możemy nadać uprawnienia do odszyfrowania naszych plików.
7. Wybrani użytkownicy, którzy dostaną uprawnienie do odszyfrowania naszych plików.
8. Nazwa wybranego pliku.
9. Postęp w pracy z wybranym plikiem, po rozpoczęciu pracy pasek będzie się ładował, aż do osiągnięcia 100%.

10. Aktualny status pliku, opis w jakiej fazie się znajduje (szyfrowanie, ładowanie nagłówka, odszyfrowywanie itd.)
11. Pasek logów, informujący o wydarzeniach i błędach w aplikacji.

Interfejs pozwalający wybrać pliku do odszyfrowywania domyślnie pozwala na wybór plików tylko o rozszerzeniu ".enc". Ma to za zadanie zmniejszyć ilość niepotrzebnych danych w interfejsie. Dodatkowe opcje z górnego menu to "About" które wyświetla dane o autorach, oraz "Settings" i "Edit" które nie działają. Zmiana głównej konfiguracji aplikacji możliwa jest jedynie z poziomu kodu, chociaż jest ona wydzielona i implementacja zmiany ustawień nie byłaby problemem.

Po zakończeniu pracy przez program, pliki zostaną umieszczone w folderze użytkownika, który znajduje się w takiej samej lokalizacji co aplikacja, a dokładniej w ścieżce /data/<nazwa_użytkownika/. Jeżeli chcemy zacząć kolejne szyfrowanie/odszyfrowywanie plików, powinniśmy najpierw kliknąć przycisk "Clear list" w celu wyszyszczenia wcześniej skończonych procesów. W przypadku gdy okaże się, że pliki istnieją już w lokalizacji docelowej, aplikacja dopisze liczbę na końcu nazwy, aby uniknąć kolizji.

1.4. Opis interfejsu logowania

Intefejs logowania otwiera się po wybraniu opcji "Login" z górnego menu. Opis poszczególnych elementów i ich działania:

1. Pole tekstowe do wprowadzenia nazwy użytkownika, podczas wpisywania będą wyświetlać się podpowiedzi, jakich użytkowników mamy dostępnych.
2. Pole tekstowe do wprowadzenia hasła użytkownika. Po wciśnięciu klawisza enter, rozpocznie się proces logowania
3. Przycisk wywołujący proces logowania.
4. Przycisk pozwalający na rejestrację w aplikacji.
5. Przycisk anulujący logowanie, który zamyka okno logowania.
6. Niewidoczne pole tekstowe, które informuje tekstowymi komunikatami o wydarzeniach i błędach.

1.5. Opis interfejsu rejestracji

Intefejs rejestracji otwiera się po kliknięciu odpowiedniego przycisku w oknie logowania. Opis poszczególnych elementów i ich działania:

1. Nazwa nowego użytkownika.
2. Hasło nowego użytkownika.
3. Powtórzenie hasła w celu werfikacji, po naciśnięciu klawisza enter, rozpocznie się proces rejestracji.
4. Przycisk wywołujący rejestrację.
5. Przycisk do anulowania operacji i zamknięcia okna.
6. Ukryte pole tekstowe informujące o wydarzeniach i błędach w procesie.

2. Szczegóły techniczne

2.1. Implementacja szyfrowania symetrycznego AES

Do szyfrowania została wykorzystana standardowa biblioteka javax.crypto, która odpowiada za szczegóły implementacyjne algorytmów, tworzenia kluczy itd. Biblioteka ta została obudowana w sposób taki, aby możliwe było szyfrowanie obiektów i strumieni. Wybrany algorytm AES oznaczony jest nazwą “AES/<tryb_blokowy>/PKCS5Padding”. Nazwa wskazuje, że pochodzi on ze standardu PKCS5 i użyto w nim Paddingu w celu zapewnienia większej ochrony. Domyślna długość klucza to 256 bitów, a ilość iteracji funkcji hashującej wynosi 2^16. Jeżeli wybrany tryb blokowy wymaga wektora inicjalizującego, jest on dostarczany w postaci 16 bajtowej za pomocą generatora SecureRandom pochodzącego ze standardowej biblioteki Java. Analogicznie tworzona jest sól potrzebna do stworzenia klucza.

2.2. Implementacja szyfrowania asymetrycznego RSA

Analogicznie jak w przypadku szyfrowania symetrycznego AES, RSA korzysta ze standardowych bibliotek Java. Wykorzystany algorytm jest oznaczony jako “RSA/ECB/OAEPWithSHA-256AndMGF1Padding” co oznacza wykorzystanie RSA z paddingiem. Długość klucza jest stała i równa 2048 bitów. W przypadku szyfrowania pliku używany jest klucz

publiczny, a podczas odszyfrowywania klucz prywatny. Do inicjalizacji używany jest SecureRandom, jednak jest to niezależne od aplikacji, ze względu na to, że jest to wewnętrzna implementacja biblioteki.

2.3. Przechowywanie kluczy prywatnych użytkowników

Klucze prywatne są przechowywane w postaci kolekcji typu mapa, gdzie kluczem jest nazwa użytkownika, a wartością zaszyfrowany klucz prywatny użytkownika. Klucz zaszyfrowany jest za pomocą hasła użytkownika, podanego podczas rejestracji. Do szyfrowania wykorzystano algorytm AES, analogicznie jak w przypadku szyfrowania plików. Tryb blokowy jest zawsze taki sam (CBC).

2.4. Nagłowki w zaszyfrowanych plikach

Każdy zaszyfrowany plik otrzymuje odpowiedni nagłówek z danymi, które pozwalają na identyfikację tego, w jaki sposób został zaszyfrowany i innych szczegółów potrzebnych do odszyfrowania go (oczywiście w wyłączniem klucza). Nagłówek ten zaczyna się od jednego bajta, który określa, czy kolejna 4 bajtowa liczba zapisana jest w postaci big czy little endian. Wartość bitu równa 1 odpowiada big endian, z kolei 0 odpowiada little endian. Następne 4 bajty reprezentują wartość liczby int, która określa długość w bajtach całego nagłówka. Informacja ta jest nam potrzebna, ze względu na to, że nagłówek w większości to ciąg bajtów zserializowanej instancji obiektu AESCipherFactory, która zawiera wszystkie jawne szczegóły użytego algorytmu i parametrów. Dzięki długości nagłówka jesteśmy w stanie odczytać cały ciąg bajtów i zdeserializować obiekt.

Ze względu na użycie serializacji, nagłówek jest praktycznie nieczytelny z poziomu danych, jednak jest to pewną formą dodatkowego zabezpieczenia. Dodatkowo dodawania kolejnych danych do nagłówka jest bardzo proste i łatwe z poziomu kodu aplikacji, dlatego zdecydowaliśmy się na takie rozwiązanie.

3. Literatura

1. http://tutorials.jenkov.com/java-cryptography/index.html
2. http://www.novixys.com/blog/java-aes-example/
3. http://www.novixys.com/blog/how-to-generate-rsa-keys-java/
4. Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C, Bruce Schneier