Tabelica projekta

|  |  |  |
| --- | --- | --- |
| Stavka | Do 28.12. |  |
| Kripto biblioteka – core | 2 | Da |
| 4 poznata algoritma | 8 | Da |
| Nepoznati algoritam – kod | 2 | XXTEA |
| Nepoznati algoritam – dokument sa opisom | 2 | / |
| Kripto servis | 4 | Da |
| Mini cloud servis | 4 | Da |
| Kripto klijent | 2 | Da |
| FSW i Buffering | 4 | Da I NE |
| Mini cloud klijent | 2 | Da |
| Ukupno | 30 |  |

# Kripto biblioteka

Definisati nezavisnu kripto biblioteku (dll) koja će imati jedinstveni intefejs za sve algoritme.

Interfejs treba da sadrži sledeće metode koje rade na nivou bajtova, i koje treba u izvedenim klasama predefinisati kako bi obezbedile funkcionalnost svakog od algoritama:

* bool SetKey(byte[] input) – postavlja ključ za kriptovanje. Vraća false ako ključ nije validan u skladu za zahtevima konkretnog algoritma
* byte[] GenerateRandomKey() – generiše random key
* bool SetIV(byte[] input) – postavlja IV za kriptovanje. Vraća false ako IV nije u skladu sa zahtevima konkretnog algoritma
* byte[] GenerateRandomIV() – generiše random IV
* bool SetAlgorithmProperties(Idictionary<string, byte[]> specArguments) – postavlja listu argumenata koji su specifični za algoritam, npr mode za simetrične kodere, dužina bloka podataka, broj rundi itd. Svi ovi parametri treba da imaju svoje default vrednosti koje će se postavljati kroz konstruktore
* byte[] Crypt(byte[] input) – prihvata niz bajtova, eventualno ga konvertuje u lokalni tip podataka, i rezultat kriptovanja vraća kao niz bajtova
* byte[] Decrypt(byte[] output) - prihvata niz bajtova, eventualno ga konvertuje u lokalni tip podataka, i rezultat kriptovanja vraća kao niz bajtova

1. **Simple substitution**

2. **XXTEA**

3. **Knapsack**

4. **SHA2**

# Kripto servis

Kripto servis je WCF servis koji treba da obezbedi kriptovanje i dekriptovanje (osim kod A5) podataka korišćenjem implementiranih algoritama kroz prethodno opisanu biblioteku.

Service treba da obezbedi pristup funkcijama iz kripto biblioteke i da klijentima omogući kriptovanje i dekriptovanje podataka.

Dakle, klijentska aplikacija treba da se poveže na kripto servis pošalje mu niz bajtova i dobije, kao odgovor, kodiran niz bajtova. Ovde obratiti pažnju na to da klijentske aplikacije mogu da uzmu niz bajtova proizvoljne veličine i pošalju na kriptovanje. Zbog toga kripto servis treba da definiše **maksimalnu veličinu** niza koji prihvata, i ka sve zahteve sa više bajtova od maksimalne definisane vrednosti **odbije**.

# Mini kripto cloud servis

Kao proširenje kripto servisa, implementirati i mini kripto cloud servis.

On treba da ima lokalno skladište podataka (baza, fajl sistem, po izboru), gde treba da čuva kriptovane fajlove koje je neko uploadovao. **Korisnik treba da uploaduje fajl** koji se na **strani servisa kriptuje**, i smešta na odgovarajuće mesto u skladište.

Ovaj servis treba da obezbedi i metoda za **ažuriranje, brisanje i vraćanje liste** svih **smeštenih fajlova**.

Poseban zahtev ovde je vezan za fajlove koji ne mogu da se iz jednog dela **uploaduju** i **downloaduju** zbog svoje veličine i ograničenja na kripto servisu za maksimalni niz bajtova. Osmisliti algoritam koji će rešiti ovaj problem.

# Kripto klijent

Kripto klijent je standardna Windows Forms aplikacija koja treba da obezbedi osnovni GUI primenjiv na sve implementirane kripto algoritme. GUI treba da uključi polje za unos teksta za kriptovanje, komponente ključa i polje gde će se prikazati kriptovani tekst. Takođe, GUI mora da podrži i dekriptovanje za svaki od implementiranih algoritama (sem za A5).

## FSW

Kripto klijent treba da bude u stanju da automatski učitava fajlove sa određene ulazne lokacije, zove sa **servisa** funkciju koja kodira sadržaj i rezultat sme[ta u odredišnu lokaciju. Kako se koji novi fajl doda na ulaznu lokaciju, tako treba da se automatski procesira od strane programa.

Komponente pogodne za ovo se zovu file system watcher komponente i dolaze kroz bilo koji framework (.NET https://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher%28v=vs.110%29.aspx, JAVA http://www.codejava.net/java-se/file-io/file-change-notification-example-with-watch-service-api, QT http://doc.qt.io/qt-5/qfilesystemwatcher.html). Ovo treba implementirati kao opciju na programu koja se može uključiti i isključiti.

Program treba da bude tako napisan da može i da dekodira kodirane fajlove.

U folder koji je definisan kao ulazna lokacija, fajlovi mogu da se snimaju i kada program ne radi. Nakon startovanja, program treba da bude u stanju da preuzme sve fajlove koji su u međuvremenu dodati i procesira ih. Program mora da podrži scenario kada su ulazna i odredišna lokacija iste.

## Buffering

U prethodnom delu je opisan sistem koji ima file system watcher mehanizam koji obezbeđuje da se svaki novododati fajl automatski kodira. Problem sa ovakvim pristupom je to što priliv novih fajlova može biti mnogo brži nego brzina kodiranja.

Kako bi se ovaj efekat umanjio treba implementirati Queue mehanizam između dela programa koji nadgleda dodavanje novih fajlova (Watcher) i dela programa koji ih šalje na kodiranje (Cypher).

U ovom scenariju Watcher treba da svaki fajl (bas svaki, bez obzira na ekstenziju i tip) dodaje u Queue.

Cypher treba da čita i kodira fajlove iz Queue-a po FIFO principu. Nakon kodiranja Cypher treba da snimi rezultat kodiranja u fajl. Cypher treba da bude u stanju da dekodira kodirane fajlove i da ih vrati u potpuno isti oblik kao pre kodiranja (ime + ekstenzija + sadržaj).

Cypher treba da otvara fajlove binarno i da ih kodira po jednom od algoritama implementiranih kroz pojedinačne zadatke.

## Mini cloud klijent

Komponentu Cipher, opisanu u prethodnom poglavlju, treba zameniti komponentom koja samo uploaduje pribavljene fajlove na mini cloud.

Takođe, mini cloud client treba da obezbedi mogućnost listanja uploadovanih fajlova, njihovog downloadovanja, dekriptovanja i prikazivanja.