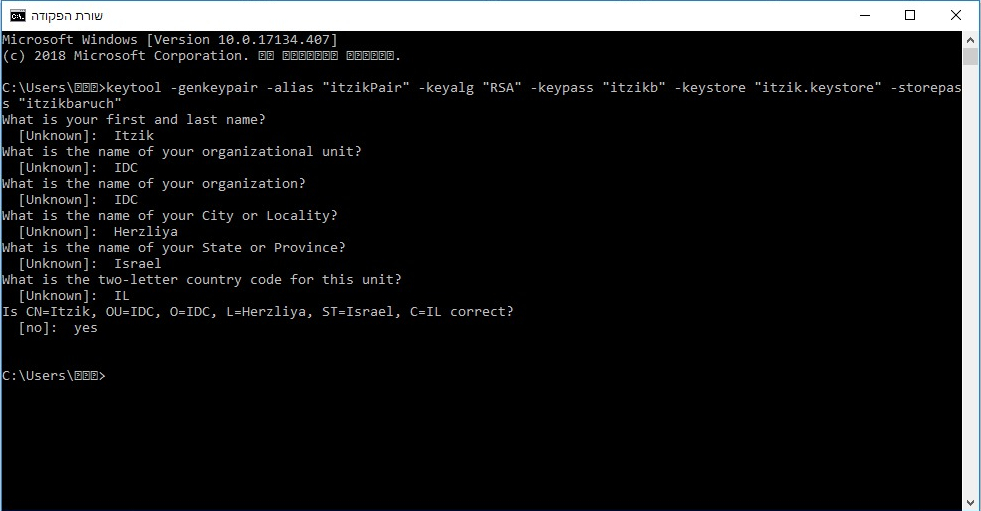
# הפקודות ליצירת ה – KeyStores

## הפקודות

### יצירת ה - KeyStores

keytool -genkeypair -alias "itzikPair" -keyalg "RSA" -keypass "itzikb" -keystore "itzik.keystore" -storepass "itzikbaruch"

keytool -genkeypair -alias "ellaPair" -keyalg "RSA" -keypass "ella13055" -keystore "ella.keystore" -storepass "ellabenatia"

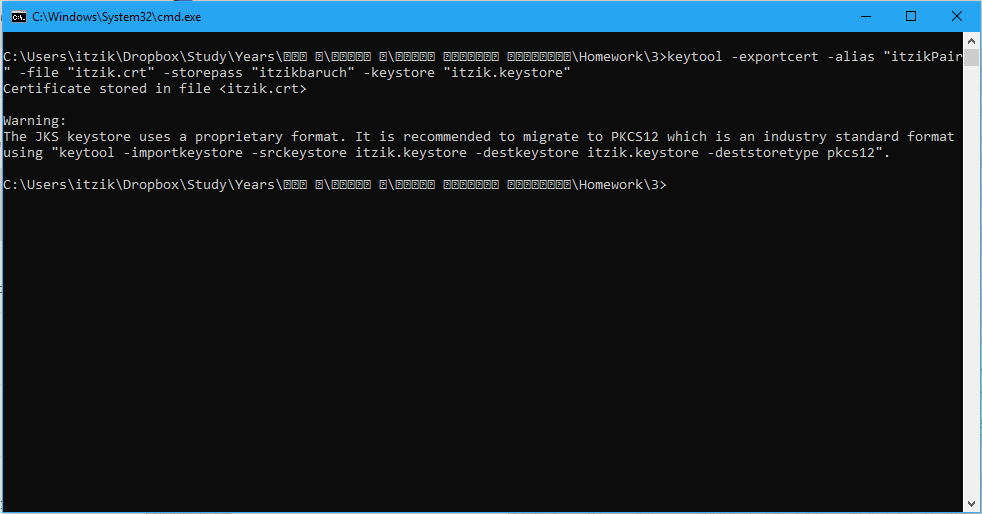


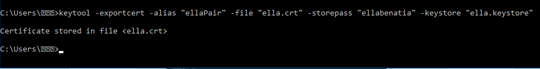
### 

### הוצאת ה – certificate מה - KeyStore

keytool -exportcert -alias "itzikPair" -file "itzik.crt" -storepass "itzikbaruch" -keystore "itzik.keystore"

keytool -exportcert -alias "ellaPair" -file "ella.crt" -storepass "ellabenatia" -keystore "ella.keystore"

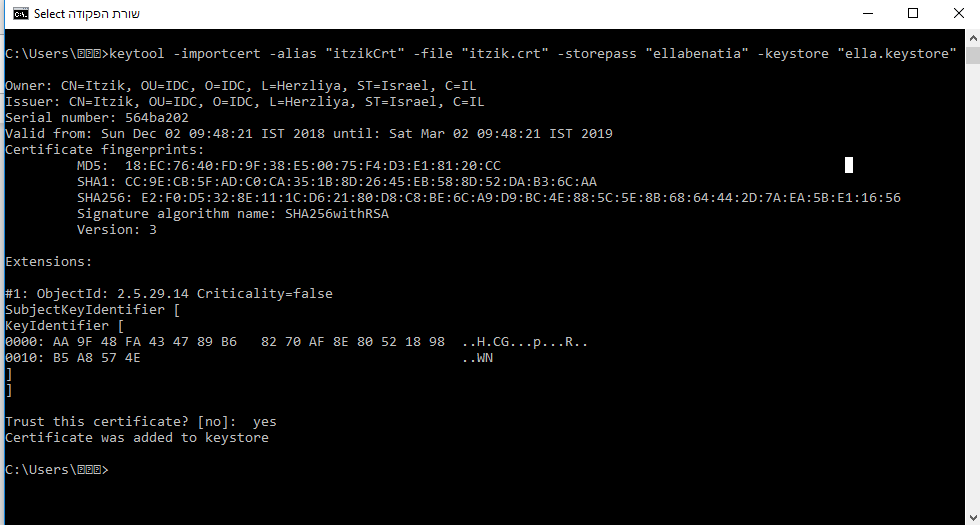


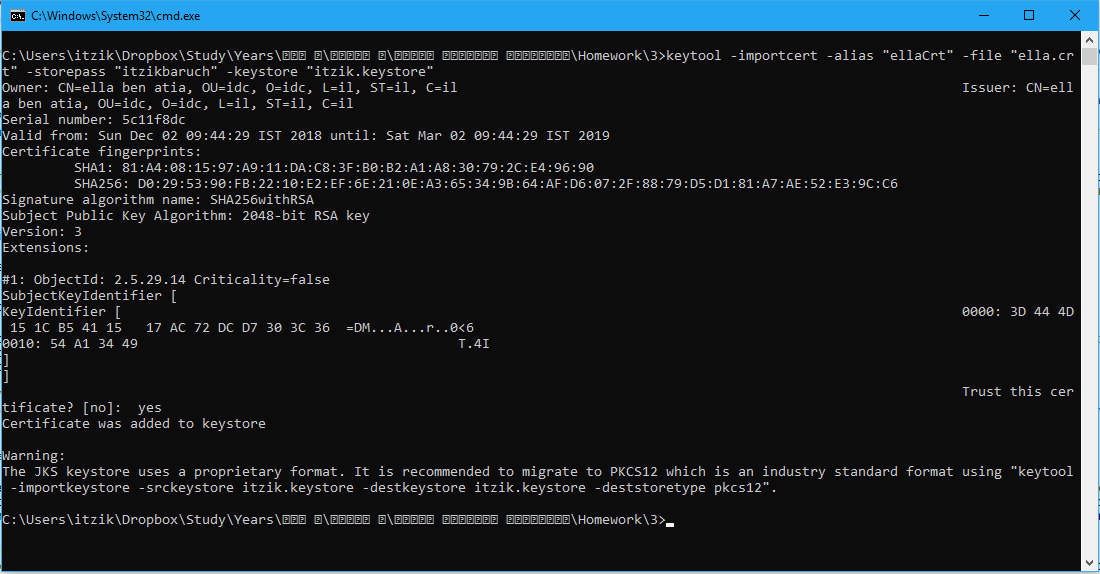


### הכנסת ה – certificate אל ה – KeyStore השני

keytool -importcert -alias "itzikCrt" -file "itzik.crt" -storepass "ellabenatia" -keystore "ella.keystore"

keytool -importcert -alias "ellaCrt" -file "ella.crt" -storepass "itzikbaruch" -keystore "itzik.keystore"





# איך להריץ את התוכנית?

כאשר מריצים את ה – jar כל הקבצים המצורפים צריכים להיות בתוך אותה תיקייה של ה – jar, וההרצה צריכה להתרחש מתוך אותה תיקייה גם כן.

הפרמטרים שצריכים להרצה (לפי הסדר):

1. שם ה – KeyStore הנוכחי שעליו פועלים.
2. הסיסמא של ה – KeyStore.
3. ה – Alias של המפתח בתוך ה – KeyStore.
4. הסיסמא של ה – Alias בתוך ה – KeyStore.
5. ה – Alias של ה – certificate.
6. באיזה מוד אנחנו: e בשביל הצפנה, d בשביל פענוח.

לכן, כדי להצפין ואז לפענח צריך להריץ פעמיים.

# איך התוכנית עובדת?

## צד המצפין

תחילת הגרלנו IV רנדומלי באמצעות SecureRandom, ולאחר מכן הגרלנו מפתח סימטרי רנדומלי באמצעות KeyGenerator.

לאחר מכן, חישבנו Hash על הקובץ באמצעות MessageDigest ועל ה- hash, בעזרת המפתח הפרטי של Itzik חתמנו על ה – hash של הקובץ הגלוי באמצעות האובייקט Singnature.

בעזרת המפתח וה – IV, בנינו אובייקט Cipher, שעובד עם האלגוריתם AES, לשם הצפנת הקובץ והכנסנו אותו לתוך CipherOutputStream כדי לקרוא את הקובץ הגלוי ולרשום את התוכן שלו מוצפן בקובץ חדש – encrypted.txt.

לאחר מכן הצפנו את המפתח הסימטרי עם המפתח הפומבי של ella שנלקח דרך התעודה הדיגיטלית שלה ששמורה אצל itzik, באמצעות RSA, בכדי לתאם אותו עם ella וכדי שאף אחד אחר לא יוכל להשתמש במפתח הסימטרי כדי להתחזות לella.

את ה – iv שיצרנו, את המפתח המוצפן ואת החתימה של הקובץ הכנסנו לתוך קובץ הקונפיגורציה כדי ש – ella תוכל לראות את ערכים אלו.

## צד המפענח

תחילה לקחנו את הערכים מקובץ הקונפיגורציה ואז בנינו אובייקט Cipher שעובד על אלגוריתם RSA, שתפקידו היה לפענח את המפתח הסימטרי המוצפן באמצעות המפתח הפרטי של ella.

לאחר מכן, לקחנו את המפתח ואת ה – IV ובנינו אובייקט Cipher חדש, שעובד על AES, שתפקידו לפענח את הקובץ. ולכן השתמשנו ב – CipherInputStream כדי לפענח את הקובץ המוצפן ולרשום את התוכן הלא מוצפן בקובץ חדש – decrypted.txt.

לאחר הפענוח, חישבנו hash שנית על הקובץ, והשתמשנו ב – Signature, עם המפתח הפומבי של itzik שנלקח דרך התעודה הדיגיטלית שלו ששמורה אצל ella, הפעם כדי לוודא שזה ה – hash שקיבלנו בקובץ הקונפיגורציה.

# למה בחרנו ב – RSA וב – AES?

AES זה האלגוריתם הסטנדרטי בשביל הצפנה סימטרית והוא מקובל בעולם, הבעיה שלו היא שצריך לתאם מפתח מראש לפני ששני הצדדים מדברים, והתיאום הזה מצריך פגישה בין הצדדים או העברת המפתח באמצעים שאינם בטוחים, וברגע שמתגלה המפתח, כל התעבורה בין 2 הצדדים גלויה.

לכן, משתמשים ב -RSA, זהו אלגוריתם הצפנה א-סימטרי אשר משתמש ב – 2 מפתחות, אחד פרטי שרק אתה אמור לדעת ואחד פומבי שכולם יודעים והם קשורים ביניהם מתמטית.

השימוש באלגוריתם א-סימטרי הוא בשביל לתאם מפתח סימטרי בשביל אלגוריתם סימטרי. אי אפשר להשתמש רק באלגוריתם א-סימטרי, הם הרבה יותר איטיים מאלגוריתמים סימטריים ולכן רק התיאום מפתח נעשה באמצעות אלגוריתם א-סימטרי ושאר התעבורה באמצעות אלגוריתם סימטרי.