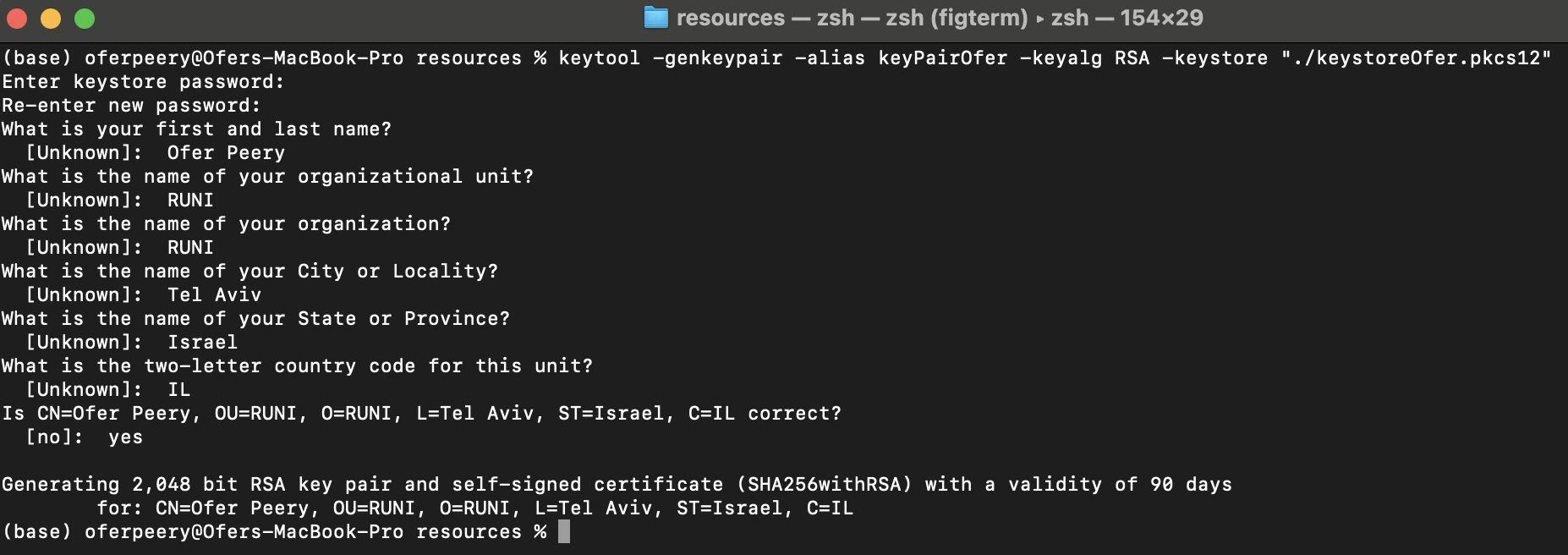
# בניית יישומים מאובטחים – תרגיל תכנותי ב-Java Crypto API

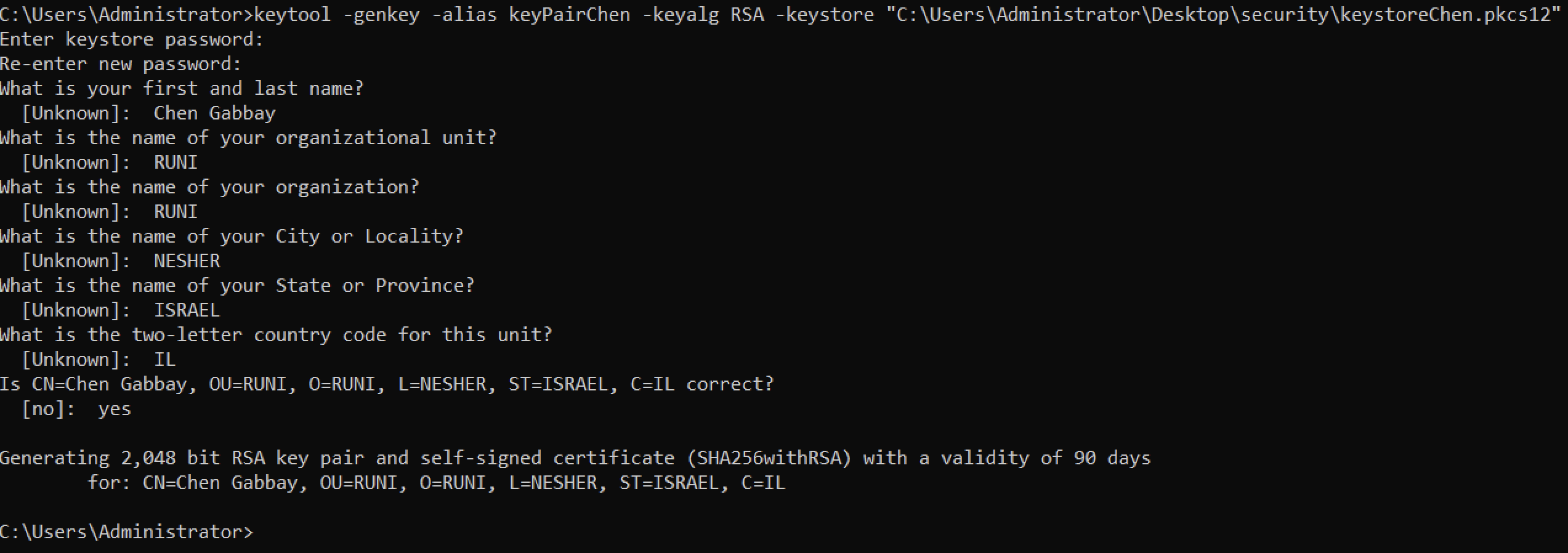
מגישים: חן גבאי 207500638 ועופר פארי 206021024

**עבודה עם keytool:**

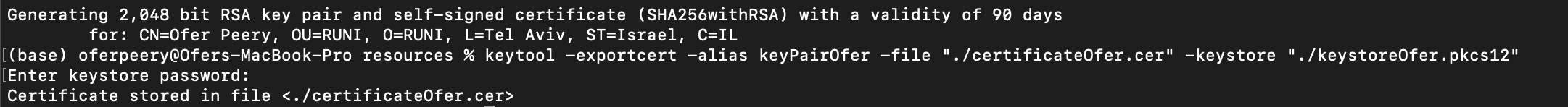
Generate a key pair for Ofer for algorithm RSA, and store it in Ofer’s keystore “keystoreOfer” of type pkcs12



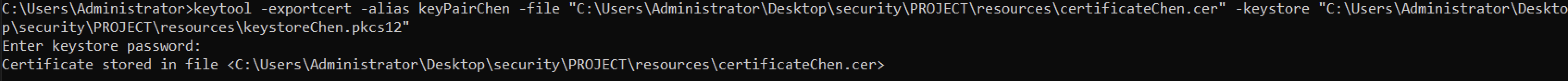
Generate a key pair for Chen for algorithm RSA, and store it in Chen’s keystore “keystoreChen” of type pkcs12



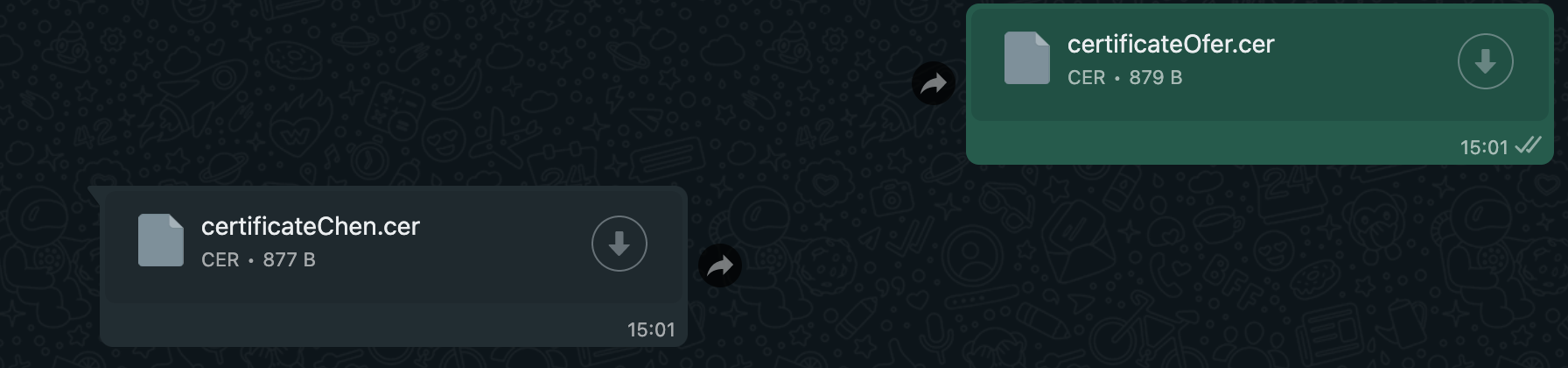
Export Ofer’s Certificate from Ofer’s Keystore



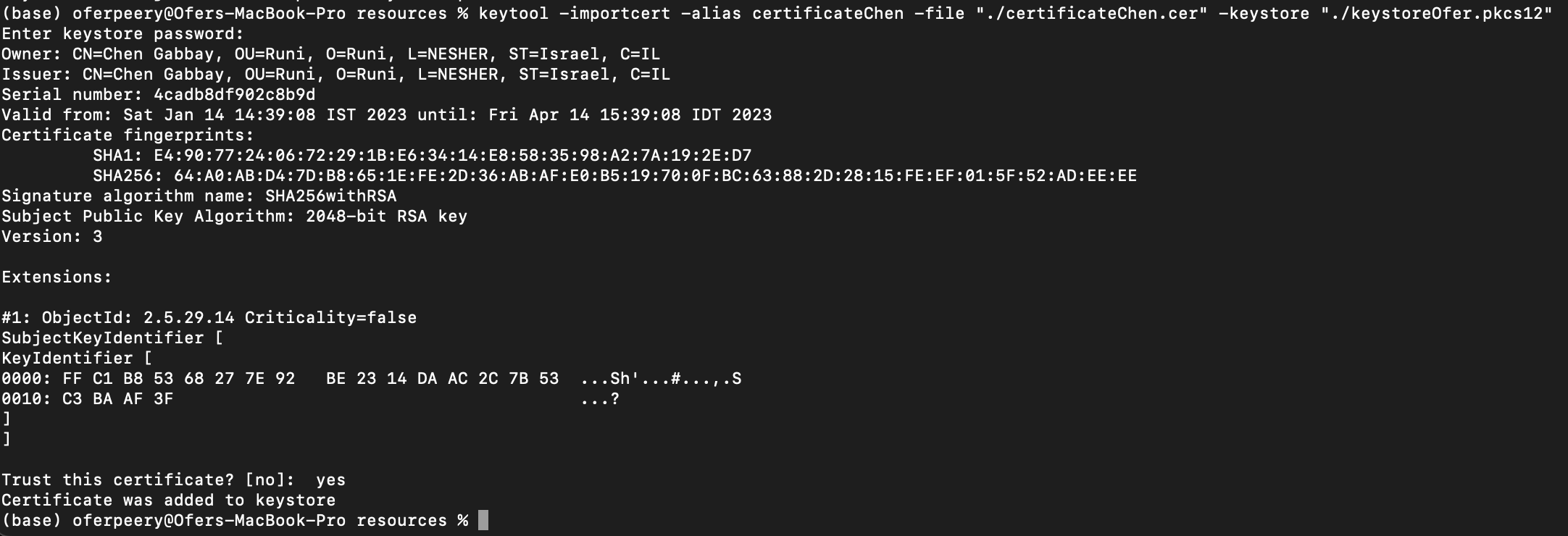
Export Chen’s Certificate from Chen’s Keystore



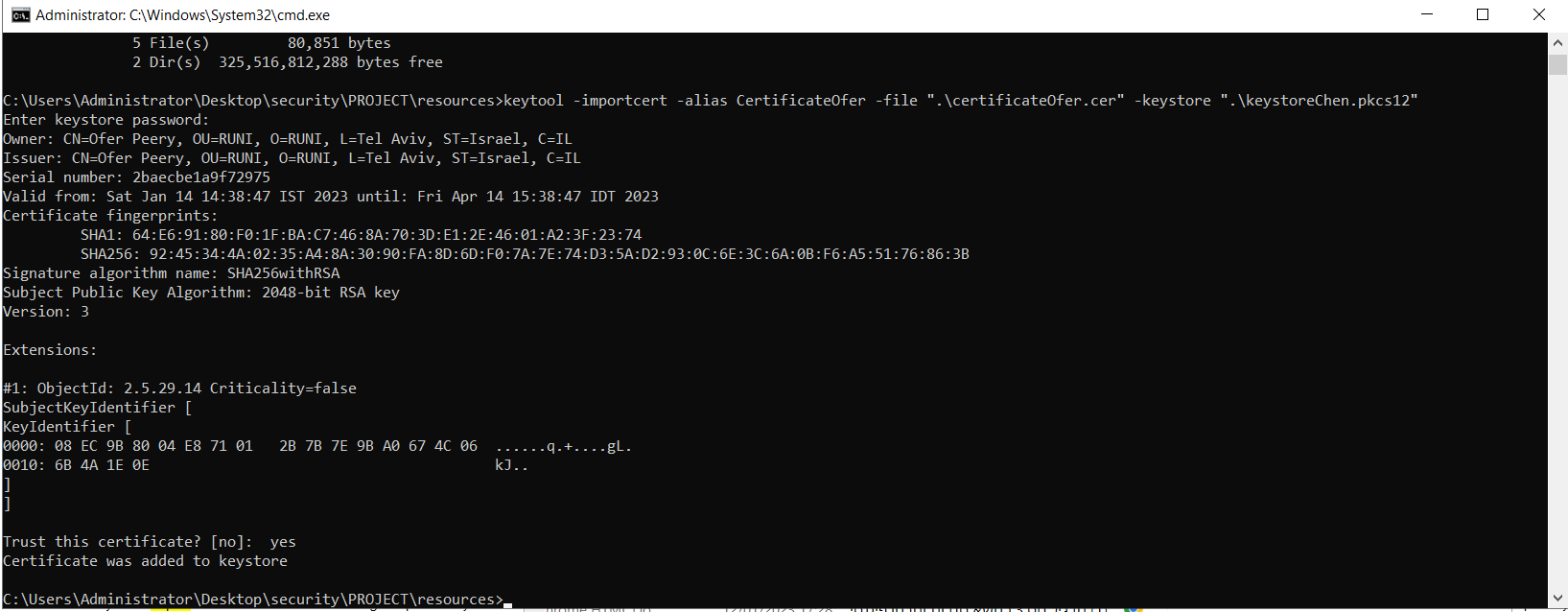
We send each other our certificates



Ofer imports Chen’s certificate to Ofer’s keystore as a trusted certificate



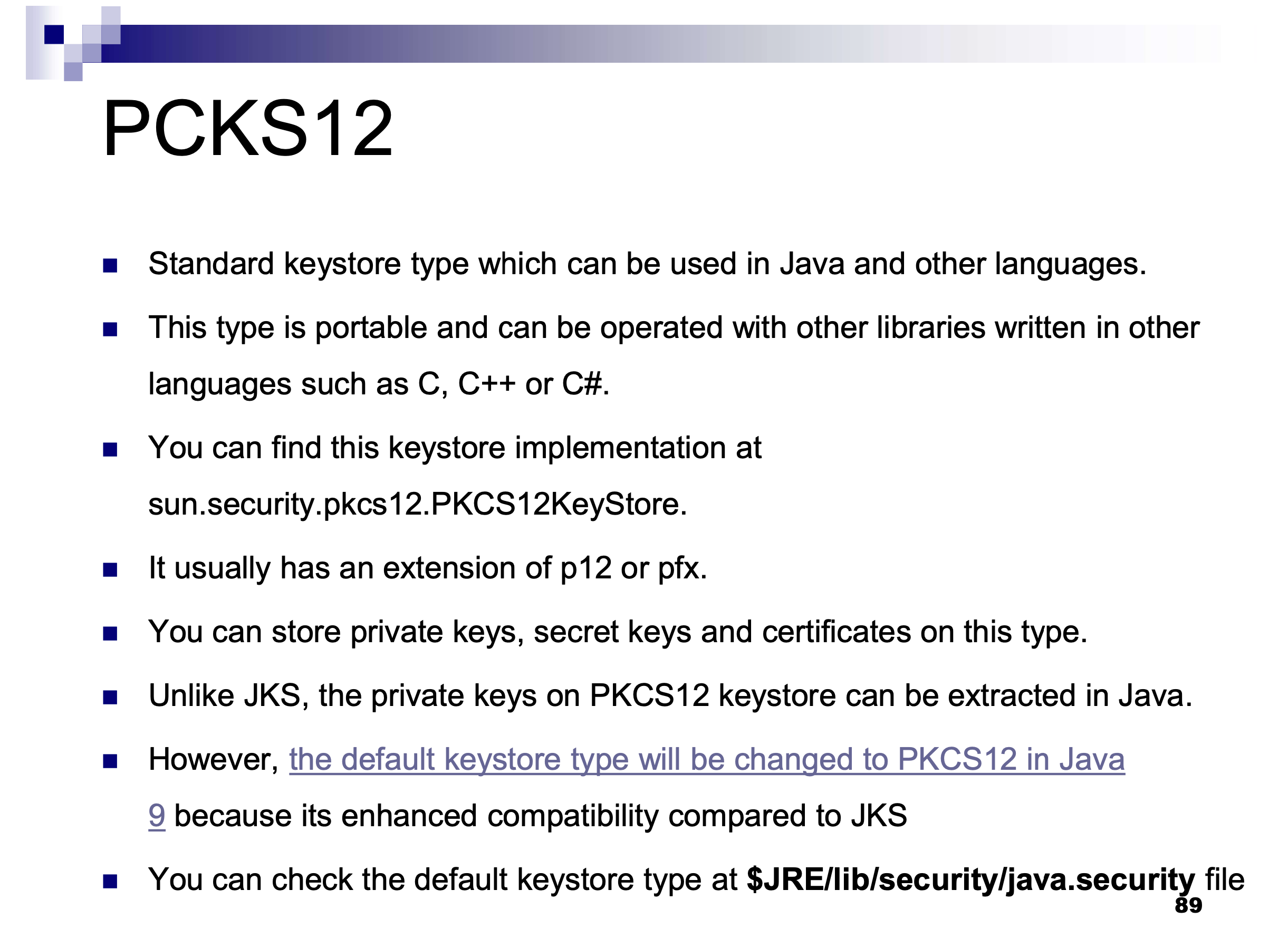
Chen imports Ofer’s certificate to Chen’s keystore as a trusted certificate



בחרנו לעבוד עם keytool מסוג pkcs12 משום שהוא אומץ בתעשייה כסטנדרט לעבודה עם keystores, בין היתר משום שעובד עם מספר שפות תכנות, בנוסף ל-Java. כמו כן, החל מ-Java 9 ה-pkcs12 אומץ גם ע״י Java כסוג ברירת המחדל עבור keystore בשל ה-compatibility שמספק בין השפות. הוא מאפשר לנו לשמור מפתחות פרטיים, מפחות סימטריים ותעודות דיגיטליות. כמו כן מאפשר לעשות extract למפתחות פרטיים, בניגוד ל-keystore מטיפוס JKS, (שבנוסף, עובד רק עם java).

[מקור המידע: שקף 89 במצגת Topic 2 – Java Crypto (JCA and JCE)]

אמנם pkcs12 מאפשר לבחור רק סיסמה אחת המשמשת הן עבור השמירה על שלמות ה-keystore והן עבור השמירה על הסודיות של המפתחות (הסימטריים והפרטיים), אך אושר לשימוש ע״י המרצה לאחר בירור. בכל אופן, בתוכנית שלנו אנחנו מאפשרים להכניס 2 קלטים אחד עבור כל סיסמה למקרה שהמשתמש יבחר להשתמש בטיפוס keystore אחר.



**הקלטים לתוכנה המצפינה:**

התוכנית מקבלת כקלט ב-command line 3 פרמטרים לפי הסדר הבא:

* הסיסמה ל-keystore השומרת על השלמות של כל ה-keystore
* הסיסמה ל-Private Key ששמור בתוך ה-keystore, השומרת על סודיות המפתח הפרטי
* הנתיב לקובץ ה-User-Input (בדוגמה שסיפקנו, שמו: encryptorUserInput.properties הממוקם בתוך התיקייה Resources שבתיקיית ה-submission).

קלטי הסיסמאות ניתנות ע״י המשתמש ב-command line משום שאנחנו מעוניינים שהן לא תשמרנה בשום קובץ. כמו כן הן לא כתובות כליטרלים בשום מקום בקוד.

בקובץ ה-User-Input נמצאים הקלטים הבאים:

|  |  |  |
| --- | --- | --- |
| **משמעות** | **הערך שבחרנו בקובץ**  **(Value)** | **שם השדה**  **(Alias)** |
| מכיל את הנתיב למיקום ה-keystore של הצד המצפין (צד A) | ./Resources/keystoreOfer.pkcs12 | keyStorePathA |
| מכיל את הנתיב לקובץ הקונפיגורציות שמקבלת תוכנית ההצפנה (הסבר על תוכנו בהמשך) | ./Resources/encryptorConfig.properties | inputConfigFilePath |
| מכיל את הנתיב לקובץ הטקסט הגלוי שאותו נצפין בהצפנה היברידית | ./plaintext.txt  כפי שהתבקשנו בהוראות המטלה | inputPlaintextFilePath |
| טיפוס ה-keystore | pkcs12  כפי שהוסבר לעיל | keyStoreType |
| ה-provider ל-keystore | SUN | keyStoreProvider |
| ה-alias של המפתח הפרטי של צד A | keyPairOfer | privateKeyAliasA |
| ה-alias של התעודה הדיגיטלית של צד B | certificateChen | certificateAliasB |
| הנתיב לקובץ שיהווה פלט לתוכן הסתר | ./Resources/ciphertext.txt | outputCipherFilePath |
| הנתיב לקובץ הקונפיגורציות שישמש את תוכנת הפיענוח | ./Resources/decryptorConfig.properties | outputConfigFilePath |

בקובץ הקונפיגורציות להצפנה, שהנתיב שלו ניתן על ידי המשתמש אל התוכנית באמצעות ציון הנתיב בקובץ ה-User-Input לעיל, על המשתמש למלא את הפרמטרים הבאים:

|  |  |  |
| --- | --- | --- |
| **משמעות** | **הערך שבחרנו בקובץ**  **+ נימוק לבחירתו(Value)** | **שם השדה**  **(Alias)** |
| שם האלגוריתם בפורמט transition להצפנה הסימטרית של הקובץ הגלוי | AES/CTR/NoPadding  בחרנו באלגוריתם AES משום שהוא הסטנדרט כיום להצפנה סימטרית, כפי שלמדנו בהרצאה. בנוסף, בחרנו במוד CTR בשל יתרונותיו שראינו בהרצאה (מקביליות בהצפנה ובפענוח, שיבוש ביט בסתר משבש רק ביט בגלוי, לא מצריך ריפוד) | symmetricEncryptionAlg |
| ה-provider לאלגוריתם ההצפנה הסימטרי הנ״ל | SunJCE | symmetricEncryptionProvider |
| גודל המפתח להצפנה הסימטרית עבור האלג׳ הנ״ל | 128  בחרנו באופציה הסטנדרטית הזו, ולא ב-256 כי 256 יקר יותר מבחינה חישובית ואנחנו לא שומרים על סודות מדינה לכן אין צורך במפתח כה חזק, כמו כן אנחנו לא רוצים לייצר מצב של  security theater  כלומר שלא יחשבו שיש לנו חולשה במקום אחר בתוכנית. | symmetricKeySize |
| האם המשתמש מעוניין שאלגוריתם ההצפנה הנ״ל ייצר את ה-IV באופן דיפולטיבי ע״י הגדרותיו.  אם כן – true  אם לא – false  (אם המשתמש בוחר אלגוריתם ומוד הצפנה שלא מצריך IV, עליו לבחור true, או לא לציין שדה זה) | false  בקובץ הדוגמה שלנו בחרנו false כדי לייצר IV לפי הפרמטרים שסיפקנו.  בתוכנית עצמה – ברירת המחדל היא true כדי שאם המשתמש לא סיפק שדה זה (ועל אחת כמה במצב שהמוד לא מצריך IV) התוכנית תשתמש בברירת המחדל של אלגוריתם ההצפנה. | flagDefaultIv |
| שם האלגוריתם ל-Secure Random ליצירת PRNG (סדרה פסודו אקראית) עבור יצירת ה-IV | SHA1PRNG  לפי התיעוד הרשמי של Oracle זה אלגוריתם ה-Secure Random הנתמך ע״י SUN.  (הערה: מחלקת המנוע SecureRandom לא תומכת ב-SHA256, ולפי התיעוד SHA1 מספיקה עבור SecureRandom, כמו כן טענה זו הוצגה בפני המרצה ואושרה לאחר בירור). | secureRandomAlg |
| ה-provider לאלגוריתם הנ״ל ל-Secure Random | SUN | secureRandomProvider |
| כמות הבתים של ה-IV | 16  שהם 128 ביטים, כפי שעובד מוד CTR | ivBytesLength |
| שם אלגוריתם החתימה הדיגיטלית (״אסימטרית״) | SHA256withRSA  סטנדרט החתימה הדיגיטלית (״האסימטרית״) כיום הוא RSA עם SHA256 כפונקציית האש קריפטוגרפית בטוחה וחזקה המספקת את התכונות הנדרשות מכזו, כפי שלמדנו בהרצאה. | signatureAlg |
| ה-provider לאלגוריתם החתימה הדיגיטלית הנ״ל | SunRsaSign | signatureProvider |
| שם אלגוריתם ההצפנה האסימטרית לשם הצפנת המפתח הסימטרי שיוגרל במהלך התוכנית | RSA  כפי שלמדנו, RSA הוא אלגוריתם ההצפנה האסימטרי הסטנדרטי כיום בתעשייה להצפנה של תוכן. כמו כן, התבקשנו להגריל זוג מפתחות א-סימטריים עבור RSA. | asymmetricEncryptionAlgForKey |
| ה-provider לאלגוריתם ההצפנה האסימטרית הנ״ל | SunJCE | asymmetricEncriptionProviderForKey |

כל ה-Providers שבחרנו הם של חברת Sun Microservices, החברה שפיתחה את Java.

**הקלטים לתוכנה המפענחת:**

התוכנית מקבלת כקלט ב-command line 3 פרמטרים לפי הסדר הבא:

* הסיסמה ל-keystore השומרת על השלמות של כל ה-keystore
* הסיסמה ל-Private Key ששמור בתוך ה-keystore, השומרת על סודיות המפתח הפרטי
* הנתיב לקובץ ה-User-Input (בדוגמה שסיפקנו, שמו: decryptorUserInput.properties הממוקם בתוך התיקייה Resources שבתיקיית ה-submission).

קלטי הסיסמאות ניתנות ע״י המשתמש ב-command line משום שאנחנו מעוניינים שהן לא תשמרנה בשום קובץ. כמו כן הן לא כתובות כליטרלים בשום מקום בקוד.

בקובץ ה-User-Input נמצאים הקלטים הבאים:

|  |  |  |
| --- | --- | --- |
| **משמעות** | **הערך שבחרנו בקובץ**  **(Value)** | **שם השדה**  **(Alias)** |
| מכיל את הנתיב למיקום ה-keystore של הצד המפענח (צד B) | ./Resources/keystoreChen.pkcs12 | keyStorePathB |
| מכיל את הנתיב לקובץ הקונפיגורציות שמקבלת תוכנית הפיענוח (הסבר על תוכנו בהמשך) | ./Resources/decryptorConfig.properties | inputConfigFilePath |
| מכיל את הנתיב לקובץ הסתר שאותו נפענח באופן היברידי | ./Resources/ciphertext.txt | inputCiphertextFilePath |
| טיפוס ה-keystore | pkcs12  כפי שהוסבר לעיל | keyStoreType |
| ה-provider ל-keystore | SUN | keyStoreProvider |
| ה-alias של המפתח הפרטי של צד B | keyPairChen | privateKeyAliasB |
| ה-alias של התעודה הדיגיטלית של צד A | certificateOfer | certificateAliasA |
| הנתיב לקובץ שיהווה פלט לתוכן המפוענח (או הודעת שגיאה אם החתימה הדיגיטלית לא התאמתה). | ./decrypted.txt | outputDecryptedFilePath |

קובץ הקונפיגורציות לפיענוח, שהנתיב שלו ניתן על ידי המשתמש אל התוכנית באמצעות ציון הנתיב בקובץ ה-User-Input לעיל, יוצר ע״י התוכנית המצפינה.

היא מכילה את כל הפרמטרים שניתנו בקובץ הקונפיגורציות להצפנה, בתוספת 3 הפרמטרים הבאים:

|  |  |
| --- | --- |
| **משמעות + הסבר** | **שם השדה**  **(Alias)** |
| התוכנית המצפינה מייצאת את ה-AlgorithmParameters שהשתמשה בהם במהלך ההצפנה. התוכנית המפענחת מקבלת אותם על מנת לפענח (כשאנחנו מאתחלים ע״י cipher.init) | algParameters |
| החתימה הדיגיטלית שהתוכנית המצפינה חתמה על הקובץ המוצפן בעזרת המפתח הפרטי של צד A. | digitalSignature |
| המפתח הסימטרי המוצפן בהצפנה האסימטרית ע״י המפתח הציבורי של צד B. נעשה בו שימוש כדי לפענח את קובץ הסתר באופן סימטרי (לאחר שנפענח את תוכן המפתח באופן אסימטרי עם המפתח הפרטי של צד B). | encryptedSymmetricKey |

על מנת שתוכנית הפיענוח תפענח את הסתר כנדרש, על פי הפרמטרים שבהם השתמשה תוכנית ההצפנה, על המשתמש (המפענח, צד B) להעביר את קובץ הקונפיגורציות כפי שהוא קיבל מהצד המצפין, אל תוכנית הפיענוח. יחד עם זאת, בזכות ה-implementation independence and interoperability שמספק ה-Java Crypto API ושבו עשינו שימוש (הרי שלא כתבנו בשום מקום בקוד provider ספציפי) המשתמש יכול לשנות את ה-providerים לאלגוריתמים כרצונו (מלבד ה-keystore, שכפי שלמדנו שאין לו דרישת compatibility בין providers שונים), או לא לציין provider מסוים ואז התוכנית שכתבנו תשתמש ב-provider ברירת המחדל המוגדר בקובץ ה-java.security.

**תיעוד קובץ הוראות ההרצה run.txt**

בקובץ 6 פקודות. כל פקודה נמצאת בשורה נפרדת. יש להריץ אותן אחת אחרי השנייה ברצף.

לכל אחת משתי התוכניות (המצפינה / המפענחת) יש 3 פקודות.

3 הפקודות הראשונות הן של התוכנית המצפינה, ו-3 הפקודות לאחר מכן הן של התוכנית המפענחת.

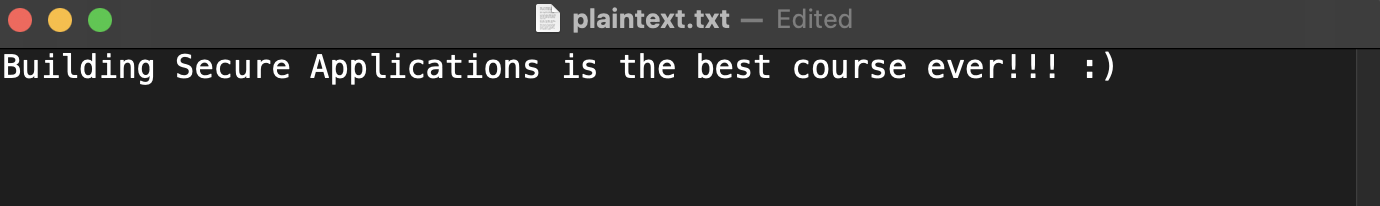
הפקודות להלן:

1. פקודה ראשונה מקמפלת את כל קבצי ה-.java הרלוונטיים לכדי קבצי .class
2. פקודה שנייה יוצרת מקבצי ה-.class יחד עם קובץ ה-manifestEncryptor.txt או ה-manifestDecryptor.txt (בהתאמה), המגדירים את ה-entry point של כל אחת מהתוכניות קובץ jar אחד לכל תכנית, עם שם האפליקציה המתאימה (EnctyptorProgram.jar או DecryptorProgram.jar).
3. פקודה שלישית מריצה את קובץ ה-jar הרלוונטי עם הקלטים הרצויים כפי שתואר לעיל.

הערה: פקודות 1 ו-2 הנ״ל (עבור כל תוכנית) נועדו כדי לאפשר קימפול של קבצי המקור למחשב של משתמש חדש בתוכניות. לאחר קימפול ויצירת קובץ jar של כל אחת מהתוכניות, ניתן להריץ את התוכנית רק ע״י פקודה 3 של כל אחת מהתוכניות, המשתמשת בקובץ ה-jar הקיים.

**דוגמת הרצה:**

נכתוב את התוכן הבא שנרצה להצפין בקובץ plaintext.txt בתיקייה submission, כנדרש:



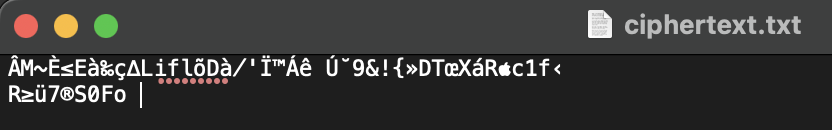
אנחנו מקמפלים, יוצרים jar ומריצים את תכנית ההצפנה עם הקלטים:

* הסיסמה ל-keystore של צד א׳ (עופר)
* הסיסמה ל-private key של צד א׳ (עופר)
* קובץ ה-User-Input - הפרמטרים של צד א׳.

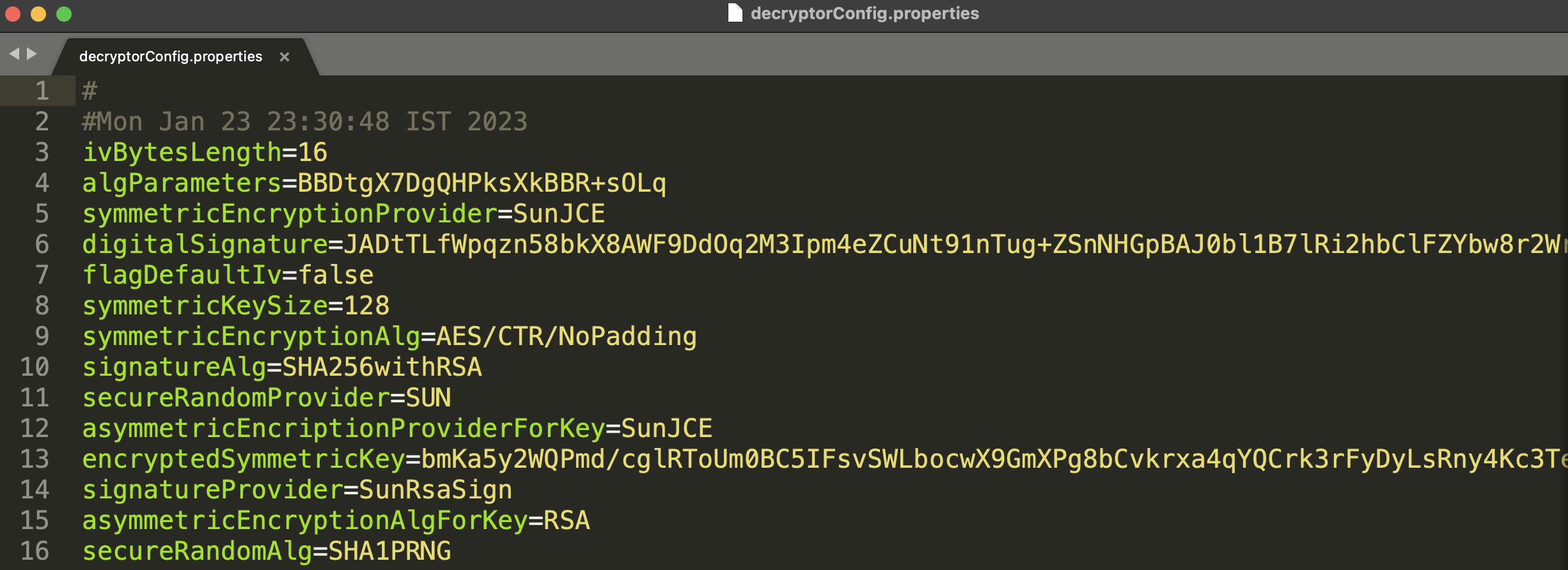
נראה כי קיבלנו פלט שהקובץ הוצפן ונחתם דיגיטלית בהצלחה.



נשים לב כי נוצר לנו בתיקייה Resources הקובץ ciphertext.txt המכיל את הסתר:



ואת קובץ הקונפיגורציות שיועבר אל התוכנית המפענחת: decryptorConfig.properties, כאשר בפרט מכיל את 3 הפרמטרים החדשים שצוינו לעיל: החתימה הדיגיטלית, המפתח הסימטרי המוצפן וה-AlgorithmParameters:



(מקודדים ב-Base64 כדי שיהיה אפשר לשלוח את הקובץ ברשת, מבלי שיחסם על ידי פרוטוקולים מסוימים שחוסמים העברת מידע בינארי).

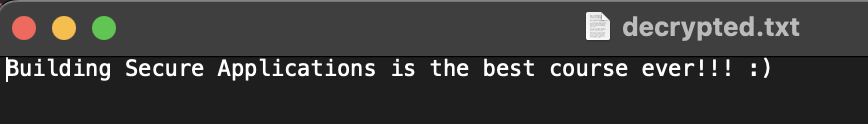
כעת, אנחנו מקמפלים, יוצרים jar ומריצים את תכנית הפיענוח עם הקלטים:

* הסיסמה ל-keystore של צד ב׳ (חן)
* הסיסמה ל-private key של צד ב׳ (חן)
* קובץ ה-User-Input - הפרמטרים של צד ב׳.

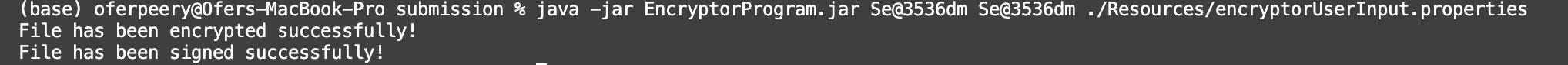
נראה כי קיבלנו פלט שווידוא החתימה הדיגיטלית ופענוח הקובץ עברו בהצלחה.



נראה כי נוצר קובץ (אם לא היה קיים, ואם היה קיים התוכן החדש ידרוס את הישן) בשם decrypted.txt בתיקייה submission, כנדרש, שכעת יכיל את הטקסט המפוענח. נשים לב כי הוא זהה לטקסט הגלוי במקור.



נשים לב כי הרצה נוספת של תוכנית ההצפנה אכן יוצרת סתר חדש:

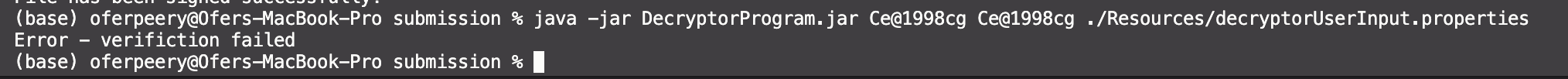




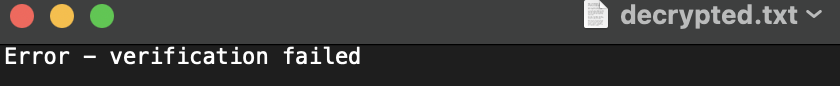
כעת נדגים שכאשר אנחנו משנים את תוכן הסתר:



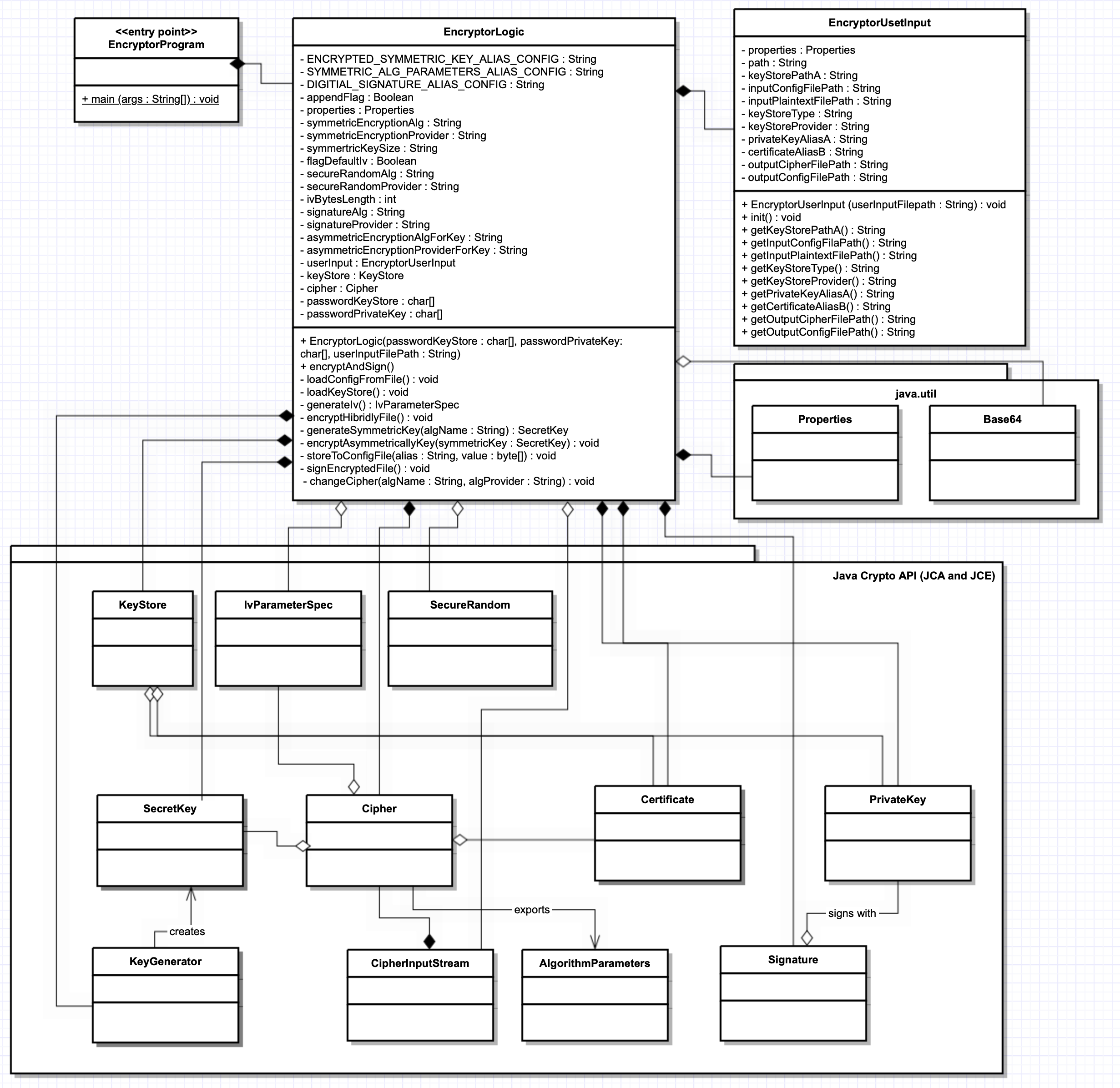
החתימה הדיגיטלית אכן תזהה ששלמות המידע נפגעה, ותחזיר הודעת שגיאה גם למסך:



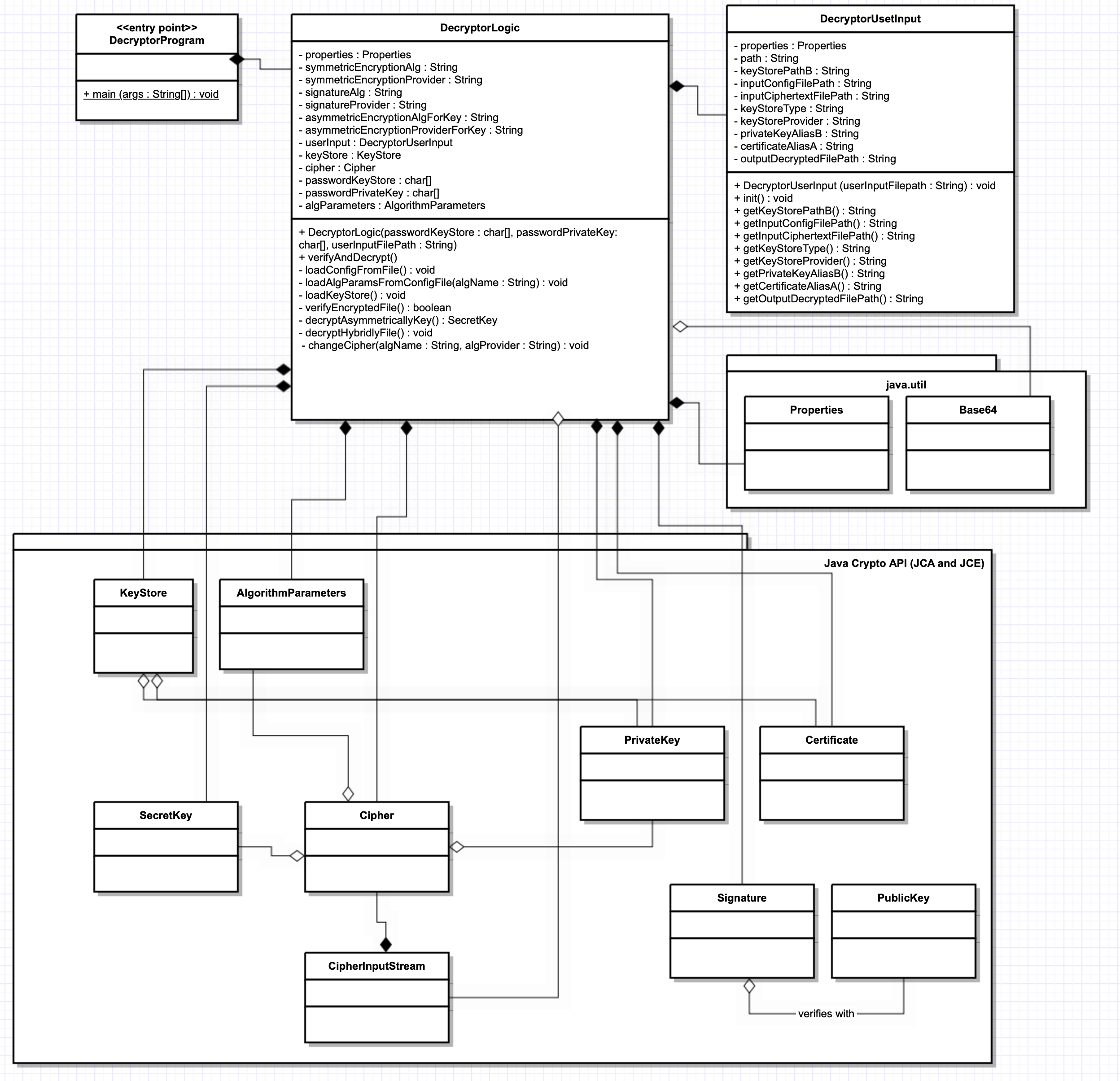
וגם לקובץ הפלט decrypted.txt בתיקייה submission:



**דיאגרמת המחלקות של התוכנה המצפינה:**



**דיאגרמת המחלקות של התוכנה המפענחת:**



**פירוט ונימוק האלגוריתמים שבחרנו**

נמצא בטבלה לעיל המפרטת את קובץ הקונפיגורציה להצפנה.

**פירוט ונימוק של ה-Class Engines עמם עבדנו בתוכנית המצפינה:**

* KeyStore – אליו אנחנו טוענים את ה-keystore של המשתמש (צד A).
  + ממנו אנחנו מחלצים את ה-private key של צד A כדי לחתום דיגיטלית על הקובץ שיוצפן.
  + בנוסף, מחלצים את התעודה הדיגיטלית של צד B (שלפני הרצת התוכנית, צד A היה צריך לטעון אותה ב-keystore שלו כ-trusted certificate) אשר מכילה את המפתח הציבורי של צד B כדי להצפין באמצעותו את המפתח הסימטרי שיוגרל במהלך התוכנית, כך שצד B יוכל לפענח אותו עם המפתח הפרטי שלו בהמשך.
* Certificate – אליו חילצנו את התעודה הדיגיטלית של צד B כמפורט לעיל.
* PrivateKey – אליו חילצנו את המפתח הפרטי של צד A כמפורט לעיל.
* Signature – באמצעותו חתמנו דיגיטלית על הקובץ המוצפן, כמפורט לעיל.
* KeyGenerator – באמצעותו אנחנו מגרילים מפתח סימטרי לצורך ההצפנה הסימטרית על הקובץ עצמו.
* SecretKey – מכיל את המפתח הסימטרי שהוגרל באמצעות ה-KeyGenerator.
* SecureRandom – באמצעותו מג׳נרטים סדרה פסודו אקראית חזקה ובטוחה לצרכים קריפטוגרפיים, ישמש אותנו לצורך הגרלת ה-IV במידה שהמשתמש בחר באמצעות קובץ הקלט flagDefaultIv == false (כלומר לא מעוניין להשתמש ביצירת ה-IV ע״י ברירת המחדל, אלא באמצעות הפרמטרים המתאימים שיספק).
* IvParameterSpec – מכיל את ה-IV שהוגרל לעיל.
* Cipher – באמצעותו נצפין:
  + נצפין בצפנה אסימטרית את המפתח הסימטרי שהוגרל לעיל, באמצעות ה-Public Key של צד B, כפי שהוסבר.
  + נצפין בהצפנה סימטרית את הקובץ הגלוי לכדי קובץ סתר, ע״י המפתח הסימטרי שהוגרל לעיל.
* CipherInputStream – בהרכבה עם FileInputStream מקובץ הגלוי, יחד עם מופע ה-Cipher הנ״ל, איפשר לנו לקרוא את הגלוי ותוך כדי להצפינו.
* AlgorithmParameters – לאחר ההצפנה הסימטרית חילצנו את הפרמטרים ששימשו להצפנה (ע״י cipher.getParameters()), וסיפקנו אותם לתוכנית המפענחת ע״י שמירתם לקובץ הקונפיגורציות שתקבל התוכנית המפענחת. (הערה: אין בקוד באופן מפורש אובייקט AlgorithmParameters משום שאנחנו שומרים אותו ישר למערך בתים ע״י cipher.getParameters().getEncoded()).

**פירוט ונימוק של ה-Class Engines עמם עבדנו בתוכנית המפענחת:**

* KeyStore – אליו אנחנו טוענים את ה-keystore של המשתמש (צד B).
  + ממנו אנחנו מחלצים את התעודה הדיגיטלית של צד A (שלפני הרצת התוכנית, צד B היה צריך לטעון אותה ב-keystore שלו כ-trusted certificate) אשר מכילה את המפתח הציבורי של צד A כדי לוודא באמצעותו את החתימה הדיגיטלית שצד A חתם על הקובץ המוצפן, לשם בדיקת השמירה על השלמות של הקובץ.
  + בנוסף, מחלצים את ה-private key של צד B כדי לפענח באמצעותו את המפתח הסימטרי שהוצפן ע״י צד A בהצפנה אסימטרית באמצעות המפתח הציבורי של B.
* Certificate – אליו חילצנו את התעודה הדיגיטלית של צד A כמפורט לעיל.
* PublicKey – אליו חילצנו את המפתח הציבורי של צד A מהתעודה הדיגיטלית של צד A.
* PrivateKey – אליו חילצנו את המפתח הפרטי של צד B כמפורט לעיל.
* Signature – באמצעותו וידאנו את החתימה הדיגיטלית על הקובץ המוצפן, כמפורט לעיל.
* SecretKey – מכיל את המפתח הסימטרי שצד A הגריל ושמר לקובץ הקונפיגורציות שנשלח אל צד B (לאחר שפוענח).
* Cipher – באמצעותו נפענח:
  + נפענח באופן אסימטרי את המפתח הסימטרי שהתקבל מצד A, באמצעות ה-Private Key של צד B, כפי שהוסבר לעיל.
  + נפענח באופן סימטרי את קובץ הסתר לכדי קובץ גלוי, ע״י המפתח הסימטרי הנ״ל (במידה שבדיקת השלמות עברה בהצלחה).
* CipherInputStream – בהרכבה עם FileInputStream מקובץ הסתר, יחד עם מופע ה-Cipher הנ״ל, איפשר לנו לקרוא את הסתר מהקובץ ותוך כדי לפענחו.
* AlgorithmParameters – מכיל את הפרמטרים ששימשו את התוכנית המצפינה להצפין, והתקבלו בקובץ הקונפיגורציות שיצרה התוכנית המצפינה. נשתמש בפרמטרים אלו לשם הפיענוח.