**פרויקט תכנות - אבטחת ישומים ברשת**

א. יצירת מפתח ציבורי ופרטי עבור צד א' באמצעות java key tool :



ב. יצירת מפתח ציבורי ופרטי עבור צד ב' באמצעות java key tool :



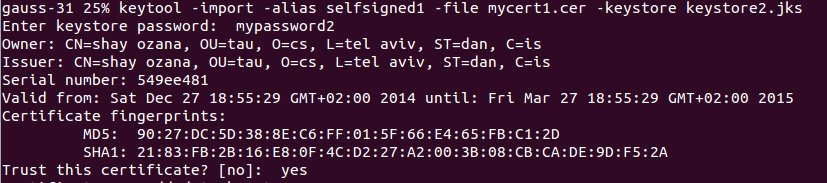
ג. הוצאת ה certificate לקובץ צד א' :



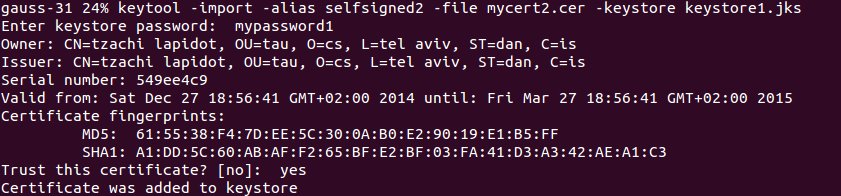
הוצאת ה certificate לקובץ צד ב' :



ייבוא ה certificate של צד א' לצד ב' :



ייבוא ה certificate של צד ב' לצד א' :



מחיקת קבצי ה certificate של שתי הצדדים : ( אין צורך בהם כי הם יובאו כבר לתוך keystore הרצוי )



**Providors and Algorithms**

השתמשנו בספקים והאלגוריתמים הבאים, לאחר שקראנו אודותם בדפי ההסבר של אורקל המופיעים ברשת , והשתמשנו בשיקולים שלמדנו בכיתה אודותיהם השיקולים שהנחו אותנו בעיקר הם הענקת אבטחה מקסימלית תוך שימוש באלגוריתים "בטוחים" ויעילים שנלמדו בכיתה . ניתן בקלות להחליף ספקים ואלגוריתמים בעזרת שינוי השדות המתאימים בקובץ קונפיגורציות שניתן לתוכנית (config\_input.cfg הינו דוגמא לקובץ שכזה, הוא מצורף לתרגיל). רשימת השדות הניתנים כקלט בקובץ הקונפיגרציות שמקבל ה-Encryptor הם כדהלן:

RandomAlgorithm=SHA1PRNG

אלגוריתים לייצור מספרים אקראיים כפי שלמדנו בשיעור(באמצעות פונקציית גיבוב) , זהו האלגוריתים המקובל ברוב סביבות העבודה ( בקרוב אמור להיות מוחלף בSHA2 בצורה סופית כפי שצויין למטה בקישור ויקוי אך למרות זאת נחשב אלגוריתים מקובל ובטוח לייצור מספרים אקראיים.

KeyEncryptAlgoProvider=SunJCE

ספק JCE כפי שהוסבר בכיתה מספק שירותי הצפנה מתקדמים והינו הרחבה של ספק JCA בעזרתו מימשנו את האלגוריתמים המצויינים לעיל בתרגיל הנ"ל בעזרתו ניתן לממש את AES , PKCS5Padding CBC , אלגוריתמים שהתמשנו בהם ומפורטים בהמשך .

Spec DigestAlgoProvider=SUN

ספק זה הגיע עם jdk 1.1 ויודע לספק אלגוריתמים שהתשמשנו בהם כגון : (SHA1) MessageDigest , SecureRandom (SHA1PRNG) כפי שמפורטים .

KeyAlgorithm=AES

תקן ההצפנה שביקשנו לממש זהו תקן הצפנה בינלאומי ונפוץ מאוד בחומרה ותוכנה נחשב תקן הצפנה בטוח

הוא משתמש באלגוריתים של צופן בלוקים [איטרטיבי](http://he.wikipedia.org/wiki/%D7%90%D7%99%D7%98%D7%A8%D7%A6%D7%99%D7%94) עם בלוק ומפתח משתנים בגודלם. האלגוריתים שלנו משתמש ב 128 סיביות.

DigestAlgorithm=SHA1

אלגוריתם הצפנה שמשתמש בפונקציית גיבוב כפי שלמדנו בשיעור זהו האלגוריתים הנפוץ ביותר מבין אלגוריתמי הצפנה גיבוב (ראה ערך ויקיפדיה <http://en.wikipedia.org/wiki/SHA-1> ומצויין שם שלמרות חסרונותיו עדיין מקובל לשימוש ע"י רוב החברות הגדולות . (כפי שמצויין שעד 2017 כבר לא ייתמך יותר ע"י חלק מהחברות הגדולות אבל בינתיים נחשב יחסית בטוח לשימוש))

SecureRandomAlgorithmProvider=SUN

מפורט לעיל.

SecretKeyAlgorithmProvider=SunJCE

בספק זה השתמשנו בהצפנת החתימה ע"י SHA1withRSA כפי שניתן לראות בשדה הבא. זוהי הצפנה שמשמשת את החתימה ( הצפנה אסימטרית באמצעות פונקציית גיבוב ) .

SignatureEncryptionAlgorithm=SHA1withRSA

אלגוריתים חתימת ההצפנה שלנו משתמש בפונקציית גיבוב SHA1 וכן באלגוריתם ההצפנה המתבקש RSA.(זהו אלגוריתם גיבוב טוב יותר מ MD5 כפי שהוסבר לעיל אשר נחשבת הצפנה פרוצה)

KeyEncryptionAlgorithm=RSA

כפי שביקשו מאיתנו בתרגול השתמשנו באלגוריתם RSA אלגוריתם הצפנה אסימטרי.

MessageEncryptAlgoProvider=SunJCE

ספק לאלגוריתים ההצפנה מפורט לעיל .

EncryptionAlgorithmForFile=AES/CBC/PKCS5Padding

אלגוריתים הצפנה(אלגוריתים (CBC AES\CBC\PJCS5Padding כפי שהתבקשנו להשתמש משתמש בספק SunJCE .

משתמש ב IV שיצרנו (באורך 16 סיביות) , ויוצר הצפנה מבוססת בלוקים כפי שראינו בשיעור להבדיל מ ECB , אלגוריתים זה לא גורם לחזרתיות ולכן למשל בהצפנת תמונה לא נזהה את הדפוס החוזר .ה IV שיצרנו הינו רנדומלי אלגוריתם זה נבחר כפי שהתבקשנו בתרגיל .

SignatureEncryptAlgoProvider=SunJSSE

זהו ספק שממש אלגוריתם הצפנת RSA בעבור חתימות זהו הינו הספק היחיד שמספק זאת עד ל JDK 1.5 כפי שמפורט בלינקים המצורפים .

המידע שלקחנו אודות הספקים שהכללנו בפרויקט ניתן לקרוא בלינק הבא:

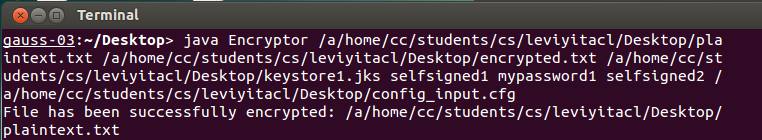
<http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider>

<http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html>

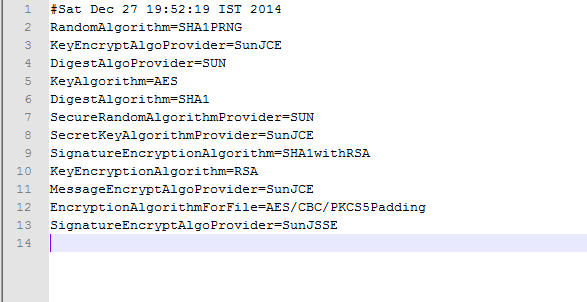
**דוגמאת הרצה ( הרצה לדוגמא שכלל הקבצים בשולחן עבודה ):**

בתחילה התוכנית מצפינה את הקובץ היא מקבלת את הארגומנטים הבאים :

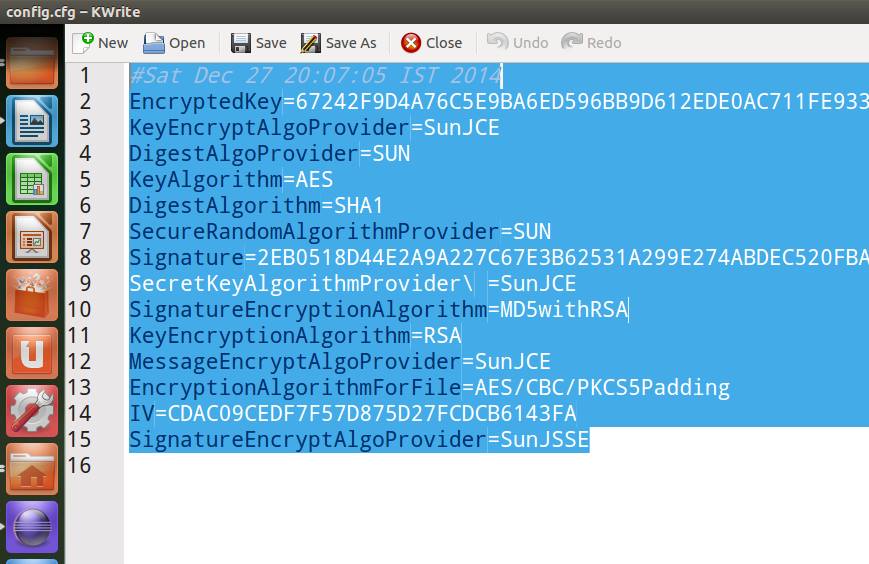
1. נתיב לקובץ ההצפנה ( plaintext.txt בדוגמא נמצא בשולחן העבודה )
2. נתיב ליצירת הקובץ המוצפן (encrypted.txt בדוגמא שולחן עבודה)
3. נתיב לקריאת ה keystore של המצפין שיוצר בסעיפים א'-ג' alias שלו , סיסמא שלו , וכן alias של הצד השני .
4. נתיב לקריאת קובץ הקונפיגורציה של המצפין .



המצפין משתמש בקובץ קונפיגורציה :



לאחר ההצפנה נוצר קובץ הקוניפגורציה הנ"ל ( באותה תיקייה שהורצה התוכנית config.cfg ) :

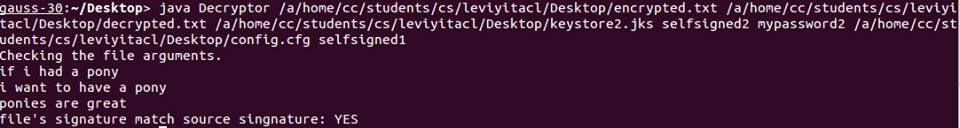


לאחר מכן הקובץ המפענח משתמש בקובץ זה ובארגומנטים הבאים ע"מ לפענח את הקובץ :

1.נתיב לקובץ המוצפן ( בדוגמא זו encrypted.txt נמצא בשולחן עבודה )

2.נתיב לקובץ התוצאה המפוענח (בדוגמא זו decrypted.txt נמצא בשולחן עבודה)

3.נתיב לקובץ ה keystore של הצד המפענח , alias שלו , וכן סיסמא שלו . וכן alias של הצד המצפין .



לבסוף התוכנית מדפיסה את התוצאה של הקובץ ואישור במידה והחתימה נכונה .