מסמך אפיון לממ״ן 16 - מטלת תכנות

הקדמה

חתימה דיגיטלית מאפשרת לחותם שהנפיק מפתח אימות ציבורי מתאים, לחתום על מסמך בצורה כזו שכל מי שיש לו גישה למפתח האימות (ויודע שהוא שייך לחותם) יכול לוודא שלא חל שינוי במסמך שהגיע ממנו בשום צורה שהיא. חתימה דיגיטלית חיונית מאוד במלחמה נגד זיופים והתחזויות. דוגמאות לשימוש מעשי בחתימה דיגיטלית אפשר לראות בדפדפן האינטרנט ובעדכוני תוכנה. דפדפן האינטרנט משתמש בחתימה דיגיטלית המונפקת על ידי רשות מאשרת כדי לזהות אתרים מפוקפקים ולהתריע בפני המשתמש על סכנות אפשריות בזמן גלישה. חברות תוכנה נוהגות לחתום על עדכוני התוכנה אותם הם מפיצים ללקוחות שלהם כדי להגן עליהם מפני שינויים זדוניים מצד גורמים עוינים בתהליך ההפצה.

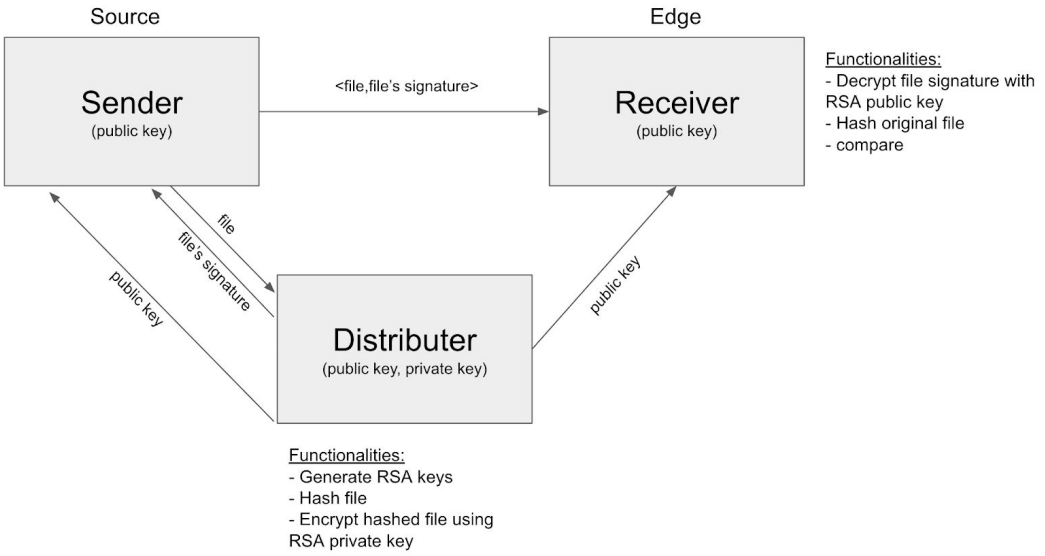
אלגוריתם ה-RSA

נניח שאליס משתמשת במפתח הציבורי של בוב כדי לשלוח לו הודעה מוצפנת. בהודעה היא יכולה לטעון שהיא אליס אך לבוב אין דרך לאמת שההודעה למעשה הגיעה מאליס מכיוון שכל אחד יכול להשתמש במפתח הציבורי של בוב כדי לשלוח לו הודעות מוצפנות.

על מנת לאמת את מקור ההודעה, ניתן להשתמש באלגוריתם RSA גם כדי לחתום על הודעה.
נניח שאליס רוצה לשלוח הודעה חתומה לבוב. היא יכולה להשתמש במפתח הפרטי שלה כדי לעשות זאת.
היא מייצרת ערך hash של ההודעה, מעלה אותו בחזקת d (מודולו n) (כמו שעושים בעת פענוח הודעה), ומצרפת אותה כ"חתימה" להודעה. כאשר בוב מקבל את ההודעה החתומה, הוא משתמש באותה פונקציית גיבוב על ההודעה המקורית שקיבל.
בוב מעלה את החתימה בחזקת e (מודולו n) (כפי שעושים בעת הצפנת הודעה), ומשווה את ערך ה- hash המתקבל עם ערך ה- hash בפועל של ההודעה. אם השניים תואמים, בוב יודע שכותב ההודעה היה בידי המפתח הפרטי של אליס, וכי ההודעה לא שונתה על ידי גורם כלשהו במהלך המשלוח.
ב-RSA ״סטנדרטי״ השולח משתמש במפתח ההצפנה הציבורי של הנמען כדי להצפין עבורו מסר כך שרק הנמען מסוגל לפענחו באמצעות המפתח הפרטי המתאים שברשותו.
השימוש ב-RSA עבור חתימה על מסרים הוא הפוך, תהליך ההצפנה נעשה ע״י המפתח הפרטי ותהליך הפענוח ע״י המפתח הציבורי, כל מי שבידו המפתח הציבורי יכול לפענח את ההודעה ולדעת שהיא מגיעה ממקור מהימן, כלומר מקור שהיה ברשותו המפתח הפרטי המתאים שבעזרתו הוא קודד את ההודעה מלכתחילה.
ניסיון ליצור הודעה עם חתימה מזויפת, משמעו פיצוח אלגוריתם ה-RSA, שבעיקרון נחשב לבעיה קשה מאוד אם אין בידך המפתח הפרטי המקורי.

ארכיטקטורה ומימוש

נממש מקרה בו נרצה להעביר קובץ המייצג חבילת תוכנה, מ-Sender ל-Receiver. כדי ש״המקבל״ יוכל להיות בטוח שלא נעשה שינוי בקובץ לאורך הדרך ע״י גורם זדוני, הוא משתמש בשירות הפצה, Distributor, המשתמש באלגוריתם RSA, על מנת להבטיח כי הקובץ ישלח בצורה מאובטחת.



מימשנו את הפרויקט כך שכל ריבוע מייצג מיקרוסרביס שונה (רץ בprocess שונה) הממש REST API, המאפשר את התקשורת בין המיקרוסרביסים השונים.

- "השולח" (Sender) - מייצג את המקור, המעוניין לשלוח קובץ ל"מקבל" (Receiver).
- "המפיץ" (Distributor) - מייצג מיקרוסרביס שהוא "איש שירות" המנהל את אופרציית הפצת הקובץ. כלומר, הוא מנהל את ההצפנה, מחזיק בידו את המפתח הציבורי והפרטי ומאפשר גישה למפתח הציבורי לכל מי שמבקש בעזרת endpoint אותו מימשנו עבור ה-REST API.
- "המקבל" (Receiver) - מייצג את היעד, המעוניין לקבל את הקובץ מה"שולח" (Sender).
- "השולח" ו-"המקבל" יכולים, בעזרת קריאת GET, לקבל את המפתח הציבורי מה"מפיץ".

כמו כן, ה"מפיץ" משתמש במחלקה הממשת את הפונקציונליות של אלגוריתם ההצפנה RSA, אותו מימשנו במחלקה בJava.

- המיקרוסרביסים השונים מדברים אחד עם השני בעזרת REST API.

*** ניתן להריץ את הפרויקט ב-3 תהליכים שונים, כאפליקציות Java, וניתן גם לארוז אותן בDocker Containerים, ולהריץ את הפרוייקט בפקודה אחת בעזרת Docker-Compose.
נשמח להבין אם זה יכול להתקבל בעבודה זו, או אין אנו נדרשים להגיע לנקודות אלו מכיוון שזה מיותר ולא רלוונטי.
כמובן שעל מריצי העבודה יהיה להתקין Docker ו-Docker-Compose על מנת להריץ את העבודה במקרה זה.

<u>אופן הרצת התוכנית</u>
בהתאם לאישור ארכיטקטורה (מבוססת מיקרו סרוויסים בקונטיינרים או לא).