המחלקה להנדסת תוכנה

פרויקט גמר – תשע"ח

שיתוף מפתחות מבוסס הסתברות עבור IoT

מאת

שרה ספרין 312548779

רעות נגר 205437874

מנחה אקדמי: דר' גיא לשם	אישור:	תאריך:
רכז הפרויקטים: דר' אסף שפנייר	אישור:	תאריך:

תקציר

במהלך השנים האחרונות, התפתחות בתחום ה"אינטרנט של דברים" (IoT) צוברת תאוצה. מכשירים אלו הופכים ליעד להתקפות. מכיוון שהמשאבים שלהם דלים לא ניתן לממש בהם פתרונות אבטחה קיימים ולכן חסרה בהם אבטחה נאותה.
פרויקט זה מתעסק במציאת פתרון אבטחה עבור רשתות קטנות של IoT. המימוש מתאפשר גם ברשתות גדולות ע"י היררכיה.
אנו מציגות פרוטוקול אבטחה חדש המורכב מחמישה שלבים:

- מציאת מאסטר
- מציאת גודל בריכת המפתחות ויצירתה
- חלוקת מפתחות
- מציאת מפתח משותף
- רשת בטוחה

עבור השלב השני בפרוטוקול הסתמכנו על מאמר מתמטי שדן בגודל מאגר המפתחות שיש ליצור על מנת שתהיה חפיפה של מפתח אחד לפחות בין כל שני צמתים ברשת בהסתברות גבוה. זאת על מנת שהרשת תהיה דינאמית, ושינויים בטופולוגית הרשת לא יפקיעו את אבטחתה.
מימשנו את הפתרון על שלושה מכשירי IoT מסוג Raspberry Pi 3, כאשר אחד הוא המאסטר והשניים האחרים הם שני צמתים ברשת.

הצהרה

העבודה נעשתה בהנחיית דר׳ גיא לשם במחלקה
להנדסת תוכנה, עזריאלי -
המכללה האקדמית להנדסה ירושלים
החיבור מציג את עבודתנו האישית ומהווה חלק
מהדרישות לקבלת תואר ראשון בהנדסה

תודות

אנו מודות
לבורא עולם
על כוח ליצור
ושכל ללמוד את הקיים

לד"ר גיא לשם
המנחה המסור
שהובילנו בבטחה
עד לסוף הטוב
ולפרויקט גמור
שחלק אתנו ידע
ולא נלאה מלהסביר
היה זמין לנו תדיר
פישט, פרש, הסביר.

תודה לבעלים
ולשאר המשפחה
אין לנו מילים
שלנו שלכם
תבורכו על העזרה
העידוד והתמיכה!

Ramat Bet Hakerem, P.O.B 3566, Jerusalem 9103501 רמת בית הכרם, ת.ד. 3566, ירושלים מיקוד
Tel. +972-2-6588000 | Fax. +972-2-6588001 | Email: info@jce.ac.il | Web: www.jce.ac.il

תוכן העניינים

מילון מונחים

- מכשיר IoT: מכשיר עם יכולות חישוב קטנות, בעל יכולת שידור אלחוטי
- צומת: מכשיר IoT שנמצא ברשת
- מאסטר/מנהיג: צומת שנבחר לתפקיד יצירת המפתחות והפצתם ברשת
- צופן RSA: מערכת הצפנה אסימטרית שמשתמשת במפתח ציבורי שגלוי לכולם ובמפתח פרטי, סודי.
- צופן AES: מערכת הצפנה סימטרית.

תיאור מסגרת הפרויקט

הפרויקט שלנו, בהנחיית ד"ר גיא לשם מאוניברסיטת בן גוריון, הינו חלק ממערך פרויקטים מחקריים בנושא ה-IoT. הוא מטפל בהיבט אבטחתי במכשירים אלו, והוא בבסיסו של כל פרויקט אחר במערך.
ליבת הפרויקט מתבססת על מאמר תאורתי, שפורסם בשנת 2002 ועוסק בחישובים מתמטיים וסטטיסטיים על מנת לאפשר שיתוף יעיל של מפתחות בין קבוצת התקנים.
המאמר מציג נוסחאות מורכבות לפתרון, ונעזרנו במתמטיקאי על מנת להבין אותם ולממשם בקוד.

מבוא

אבטחת מידע

אבטחת מידע (באנגלית: Information Security) היא ענף העוסק בהגנה של מידע ומערכות מידע מפני כל גישה למידע שאינה ע"י גורמים מאושרים, לכך עליה לספק את שלושת הבאים: סודיות, שלמות וזמינות של המערכות והמידע בהן. אבטחת מידע היא תחום מתקדם מאוד בימינו. האפשרויות הרבות שעומדות לתוקפים של מערכות ממוחשבות להזיק בתחומים כמו פרטיות, פיננסים וביטחון מניעה את האנושות להגן ככל האפשר על מערכות אלו. מושקעים משאבים רבים בעיצוב המערכות, קידוד ותחזוקה שוטפת כדי לאתר פרצות אבטחה מוקדם ככל האפשר. עם התקדמות היכולות של התוקפים נדרשות לעתים פריצות דרך מצד קהילת המפתחים ולכן אבטחת מידע זהו תחום שמושקעים בו כסף ומשאבים רבים במיוחד על מנת לחקור ולגלות דרכי הגנה חדשות ויעילות יותר להתמודדות מול התוקפים[1].

כוח יישומי חשוב באבטחת המידע היא קריפטוגרפיה. זהו ענף במתמטיקה ומדעי המחשב העוסק במחקר ופיתוח שיטות אבטחת מידע ותקשורת נתונים, ומייצר שיטות למימוש בפועל של מושגי אבטחת המידע. הגנה זו מבוצעת על ידי הצפנת המידע בעזרת פונקציית הצפנה כלשהי, ושימוש במפתח (Key) שהוא רצף תווים סודי שאינו ידוע לתוקף. הצופן (Cipher) הוא כתב הסתר שמתקבל על ידי הפעלת פונקציית ההצפנה על טקסט הקלט. פונקציה הצפנה אידיאלית היא כזו שבהינתן הצופן- הפלט, לא ניתן להסיק ממנו מידע על הקלט.

צופן RSA

RSA היא מערכת הצפנת מפתח ציבורי דטרמיניסטית מעשית הראשונה שהומצאה והיא עדיין בשימוש נרחב במערכות אבטחת מידע מודרניות , תקשורת מחשבים ומסחר אלקטרוני . ב־RSA , כבכל מערכת מפתח ציבורי, מפתח ההצפנה אינו סודי והוא שונה ממפתח הפענוח שנשמר בסוד, על כן היא נקראת אסימטרית. האסימטריה ב־ RSA נובעת מהקושי המעשי שבפירוק לגורמים של מספר פריק שהוא כפולה של שני ראשוניים גדולים, שהיא בעיה פתוחה בתורת המספרים.

השימוש: השולח משתמש במפתח ההצפנה הציבורי של הנמען כדי להצפין עבורו מסר כך שרק הנמען מסוגל לפענחו באמצעות המפתח הפרטי המתאים שברשותו.

אלגוריתם RSA נחשב איטי יחסית ועל כן אינו מתאים להצפנה ישירה של מידע בכמות גדולה. לכן RSA משמש לשיתוף והעברה של מפתח סימטרי סודי אשר בתורו משמש להצפנה מהירה של המידע עם אלגוריתם סימטרי כמו AES[2].

צופן AES

AES (באנגלית: Advanced Encryption Standard) הוא צופן בלוקים סימטרי ומהיר במיוחד שאומץ על ידי המכון הלאומי לתקנים וטכנולוגיה (NIST) של ארצות הברית כתקן הצפנה רשמי שהתקבל בעולם כולו, להצפנת נתונים מאסיבית. אלגוריתם AES נמצא בשימוש מעשי נרחב בכל העולם הן בתוכנה והן בחומרה וידוע כאלגוריתם בטוח. [3]

גודל מפתח ההצפנה ב-RSA של 2048 ביט מקביל ברמת האבטחה לגודל מפתח של 128 ביט ב-AES.

מכשירי IoT

בעולם הטכנולוגי כיום קיימת מגמה מואצת להפוך כל מכשיר המכיל שבב אלקטרוני לבעל יכולת חיבור לרשת. כבר כיום ניתן לראות מכשירים מכל הסוגים שקיימת בהם אפשרות חיבור לאינטרנט. לדוגמה: מצלמות, מדפסות, שלטי מזגן ועוד. החזון לחבר כל מכשיר חשמלי לרשת נובע מן הרצון שמכשירים כאלו ישדרו למכשירים ומערכות סביבם את המידע שבידם, ויקבלו מידע מן הרשת לגבי אירועים ותרחישים שונים. הודעות אלו ישדרגו את יכולותיהם ופעילותם והם יוכלו להתנהג בצורה יותר "חכמה", להסיק מסקנות ולפעול אוטומטית על פיהם ללא התערבות אנושית. לדוגמה, כאשר השבב של מערכת החלונות המותקנת ב"בית חכם" קולט שידור ממערכת המיזוג על הפעלת המזגן, הוא יודע לסגור את החלונות המתאימים באופן אוטומטי כדי לייעל את פעולת המיזוג.

מגמה זאת נקראת: "אינטרנט של דברים" (IoT- Internet Of Things). זוהי רשת של חפצים פיזיים, או "דברים" המשובצים באלקטרוניקה, תוכנה וחיישנים המאפשרים תקשורת מתקדמת בין החפצים ויכולות איסוף והחלפת מידע. רשת זו צפויה להוביל לאוטומציה בתחומים רבים. כיום ישנה התפתחות נרחבת בתחום ה IoT, למשל "הבית החכם", שבו כל המכשירים מחוברים לרשת וניתן להפעיל אותם בשלט רחוק ולתאם בין פעולותיהם. תחום הIoT צפוי לגלגל מחזור של כ20 ביליון דולר בשנת 2020, על פי הודעת חברת 'סיסקו' העולמית, ומושקעים בו משאבים רבים לפיתוח מצד החברות בתחום החומרה והתוכנה.

לקראת השינוי הזה יידרש שיפור גם ברמות האבטחה המקובלות כיום בקרב מכשירים כאלו, שעצם חיבורם לרשת חושף אותם להתקפות חיצוניות והם עלולים להוות טרף קל לתוקף. בפרויקט זה נתמקד בחקירת פתרונות אבטחה, בפרט בשימוש בהצפנת RSA, עבור תחום ה"אינטרנט של הדברים" ויצירת פתרונות אבטחה ייעודיים עבורו המתחשבים בחזקות והחולשות של המוצרים הקיימים בשוק[4].

תיאור הבעיה

בהבנה של הצורך הבסיסי באבטחה ראויה למידע המשתמש והכרה במגבלות המשאבים של מכשירים אלקטרוניים קטנים, ניתן להבחין במספר בעיות ביישום אבטחה במכשירי IoT:

- מכיוון שמכשירי הIoT הינם מכשירים זולים וקטנים לרוב, ומאופיינים ביכולות עיבוד חלשות, ומשאבים נוספים דלים כמו: זיכרון, שידור וחישה, לכן להטמיע בהם יכולות אבטחה מתקדמות כמו שקיימות במערכות מחשבים גדולות זוהי משימה קשה וכמעט לא נתמכת מבחינת החומרה של ה IoT.
- מכיוון שהשימוש ב IoT הוא כמכשירים שיש להם בדרך כלל תפקיד עיקרי ייעודי והחיבור לרשת רק מוסיף להם יתרון, הם לא נתפסו עד היום בציבור כבעלי ענין עבור תוקפים. אך עם השימוש הגובר בהם מיום ליום נוצלו פרצות האבטחה שבמכשירים מסוג זה לתקיפת מערכות רגישות, כמו מצלמות אבטחה, מדפסות ועוד. לכן נדרש פתרון שיספק אבטחה הולמת כנגד ניסיונות פריצה אפשריים.
- פתרונות האבטחה שכן מיושמים כיום בתחום ה IoT הינם ייעודיים עבור מכשיר מסוג מסוים, ומוטמעים על ידי היצרן. לא נלקחת בחשבון התאמה לשוק המוצרים הכולל של IoT, ולכן פתרונות אלו אינם מתאימים מבחינת תצורה, מגבלות משאבים וייעוד, עבור כל התחום.
- רשתות מכשירים אלו נפרשות על פני אזורים גאוגרפיים נרחבים והם מפוזרים בתפוצה גבוהה. נתון זה מקשה על יכולת ניהול וניטור מצב האבטחה ברשת על ידי גורמים חיצוניים ונדרש יישום פנימי ועצמאי של עקרונות אבטחה.

על מנת לפתור את הבעיות הנ"ל נדרש פיתוח חדשני שיענה על דרישות האבטחה הגבוהות בשוק, יחד עם המגבלות המאפיינות את מכשירי הIoT.

נבהיר שפרויקט זה מתמקד ברשתות מקומיות עצמאיות של IoT ולא כאלה שמחוברות לאינטרנט. דוגמאות לרשתות כאלה יהיו: רשת של מיקרופונים השתולים ברחבי עיר שחלופת מידע ביניהם חיונית לזיהוי מקום התרחשות של אירוע ירי לדוגמה, או מכשירים המזהים מיקום של חיילים ומונעים ירי על כוחותינו בעזרת תקשורת פנימית ביניהם. כמובן שתשדורות רגישות אלו צריכות הגנה מפני מאזינים פוטנציאליים ולכך הפרוטוקול שלנו נותן מענה.

תיאור הפתרון

קיימות שתי מגמות קוטביות בניסיון להגיע להסכמה על מפתח משותף בין שני צמתים על מנת שישתמשו בו להצפנת ההודעות ביניהם בצופן סימטרי ומהיר:

האחת היא לספק מפתח יחיד לכל ההתקנים ברשת ואתו יצפינו את כל ההודעות ביניהם. גישה זו קלה למימוש ואינה דורשת משאבים רבים. אולם, קיימת בה פרצת אבטחה חמורה שברגע שנפרץ מכשיר אחד ברשת- המערכת כולה אינה בטוחה.

מגמה שניה היא ליצור "לחיצת יד" בין כל זוג התקנים ברשת וכך יסכמו על מפתח משותף שישמש אותם בתקשורת ביניהם. זוהי גישה שמספקת את האבטחה הטובה ביותר מכיוון שחשיפת התקן ברשת משאירה את כל שאר התקשורת מוגנת. זוהי אבטחה שלמה, אך מחירה מתבטא באיבוד של משאבים רבים ויקרים של התקנים חלשים אלו ופוגם ביעילות המערכת.

הפתרון שאנו מציגות, מנסה לשלב בין שתי הגישות לעיל: לקבל מקסימום אבטחה במינימום תקורה של שימוש במשאבי ההתקנים. אנו מעוניינות לשתף מפתחות בין ההתקנים ברשת באופן הסתברותי, כך שבהסתברות גבוהה מספיק תהייה חפיפה בין קבוצות מפתחות שמחזיקים כל שני התקנים.

על מנת שהתהליך יהיה דינאמי, עצמאי ולא יזדקק לניהול חיצוני הוא ייעשה רק ע"י המכשירים בתוך הרשת.

פרוטוקול אבטחה חדש עבור רשת התקני IoT

על ידי מערכת חדשה לניהול מפתחות המבוססת על שיתוף מפתחות הסתברותי בין ההתקנים.

להלן הפרוטוקול:

1. יצירת קבוצה של התקני IoT עם תקשורת ביניהם והגדרת מנהיג לקבוצה.
2. המנהיג מייצר מאגר מפתחות. המנהיג צריך לקבוע מה יהיה P, גודל המאגר, כאשר P מתבסס על המשתנים להלן. המנהיג קובע גם את רמת ההבטחה שתהיה חפיפה של מפתח אחד לפחות בין כל שני התקני IoT במערכת.

<u>האלגוריתם למציאת גודל המאגר[5]:</u>

M - גודל הזיכרון הפיזי (לדוג' 32M).

n - מספר התקני ה-IoT ברשת.

n' - גודל ה'שכונה' של צומת ברשת האלחוטית. 'שכונה' זוהי קבוצת התקנים שיכולה להיות ביניהם תקשורת ישירה, כלומר שנמצאים בטווח השידור הפיזי אחד של השני.

Pc - ההסתברות שיש לפחות שני התקני IoT בשכונה שלהם לפחות מפתח אחד משותף באופן ודאי.

k - גודל תת הקבוצה של מפתחות שיחזיק כל צומת ברשת. k מוגבל ע"י M, שכן מכשיר לא יכול להכיל קבוצת מפתחות הגדולה מהזיכרון הפנוי שלו.

p' - ההסתברות שלכל שני צמתים בשכונה קיים מפתח משותף.

- חשב את c, כאשר c קבוע:

$$P_c = e^{e^{-c}} \rightarrow c = -\ln(\ln(P_c))$$

- חשב את p: ההסתברות ששני צמתים מחוברים ישירות:

$$p = \frac{\ln(n)}{n} - \frac{c}{n}$$

- חשב את d: הערך המצופה של הצומת:

$$d = p(n-1)$$

- חשב את p' :

$$p' = \frac{d}{n'-1}$$

- בהתחשב בערך של p', השתמש במשוואה הבאה כדי לחלץ את הערך של P:

$$p' = 1 - \frac{(1-\frac{k}{P})^{2(P-k+1/2)}}{(1-\frac{2k}{P})^{2(P-2k+1/2)}}$$

בהינתן P גודל המאגר כמו שחושב באלגוריתם המוצג בשלב 2, המנהיג צריך לייצר מאגר של מפתחות בגודל P ,ולכל מפתח במאגר להוסיף מספר סידורי (index).

3. המנהיג שולח לכל חבר ברשת תת קבוצה שונה של מפתחות בצורה מוצפנת.
4. כל זוג צמתים ברשת מוצאים את המפתחות המשותפים להם.
5. כאשר צומת רוצה לתקשר עם צומת אחר התקשורת תהיה מוצפנת עם המפתח המשותף.

הפתרון שלנו מאפשר להימנע מניהול האבטחה באמצעים חיצוניים לרשת, ממומש כולו ע"י ההתקנים הנמצאים ברשת, ומספק אבטחה טובה מספיק עבור התקשורת בין המכשירים. הפתרון שלנו גמיש לשינויים בטופולוגית הרשת בכך שהתקן יכול להצטרף לרשת בכל זמן נתון והמנהיג ישלח לו תת קבוצה חדשה של מפתחות, וכן התקן שעוזב את הרשת לא חושף את כל מפתחות ההצפנה וניתן להשתמש בכל שאר המפתחות שלא נחשפו.

תיאור הכלים המשמשים לפתרון

בפרויקט עבדנו עם מכשירי Raspberry Pi 3, מכשיר בעל קלט ופלט אלחוטיים ויכולת עיבוד קטנה על מנת לבדוק וליישם את האלגוריתם.

הכתיבה נעשתה בשפת Python , הספריות העיקריות שהשתמשנו בהם:

- socket ליצירת תקשורת
- pycryptodome למימוש הצפנות בRSA ובAES
- threading בשביל מקביליות
- math, numpy, scipy לפתירת הנוסחאות המתמטיות שנדרשו על מנת למצוא את גודל הבריכה שעל ה"מאסטר" לייצר.

ארכיטקטורת המימוש

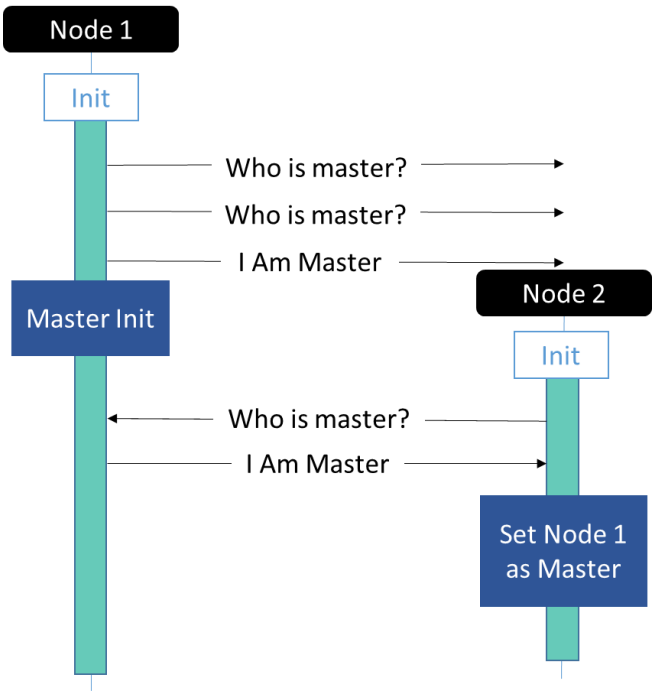
התמקדנו בתכנון קוד פשוט ומודולרי שיבצע את פעולות המאסטר והקליינט באותו קובץ הרצה, תוך שמירה על קיום מופע אחד בלבד של "מאסטר" ברשת. בנוסף הקפדנו שהתוכנה תהיה גנרית ותוכל לרוץ על מגוון של מערכות הפעלה והתקנים.

ארכיטקטורה של שלבי הפרוטוקול:

1. אתחול ומציאת מאסטר

מיד כשמכשיר נכנס לרשת הוא שולח הודעת broadcast ברשת: IS_THERE_MASTER ומחכה פרק זמן לתשובה:

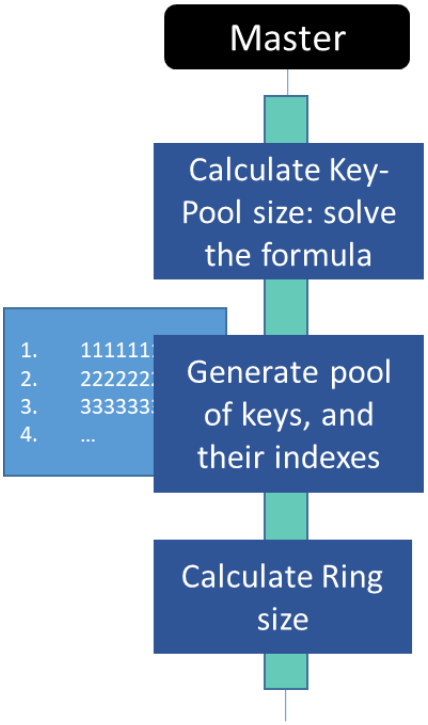
- אם קיבל הודעה I_AM_MASTER - יעבור למצב MASTER_FOUND, ויפסיק בחיפוש המאסטר. מעתה יאזין רק להודעות מהמאסטר, על מנת לקבל את המפתחות ושאר התהליך עד לסיום הקמת רשת מאובטחת.
- במקרה ולא קיבל תשובה (פעמיים, שמא קרתה תקלה ברשת) הוא מכריז על עצמו כמאסטר ועובר למצב MASTER_FOUND. הוא שולח הודעת broadcast I_AM_MASTER. אם יקבל הודעת IS_THERE_MASTER הוא ישלח הודעה פרטית לאותו מכשיר I_AM_MASTER.



2. חישוב גודל מאגר המפתחות ויצירתו

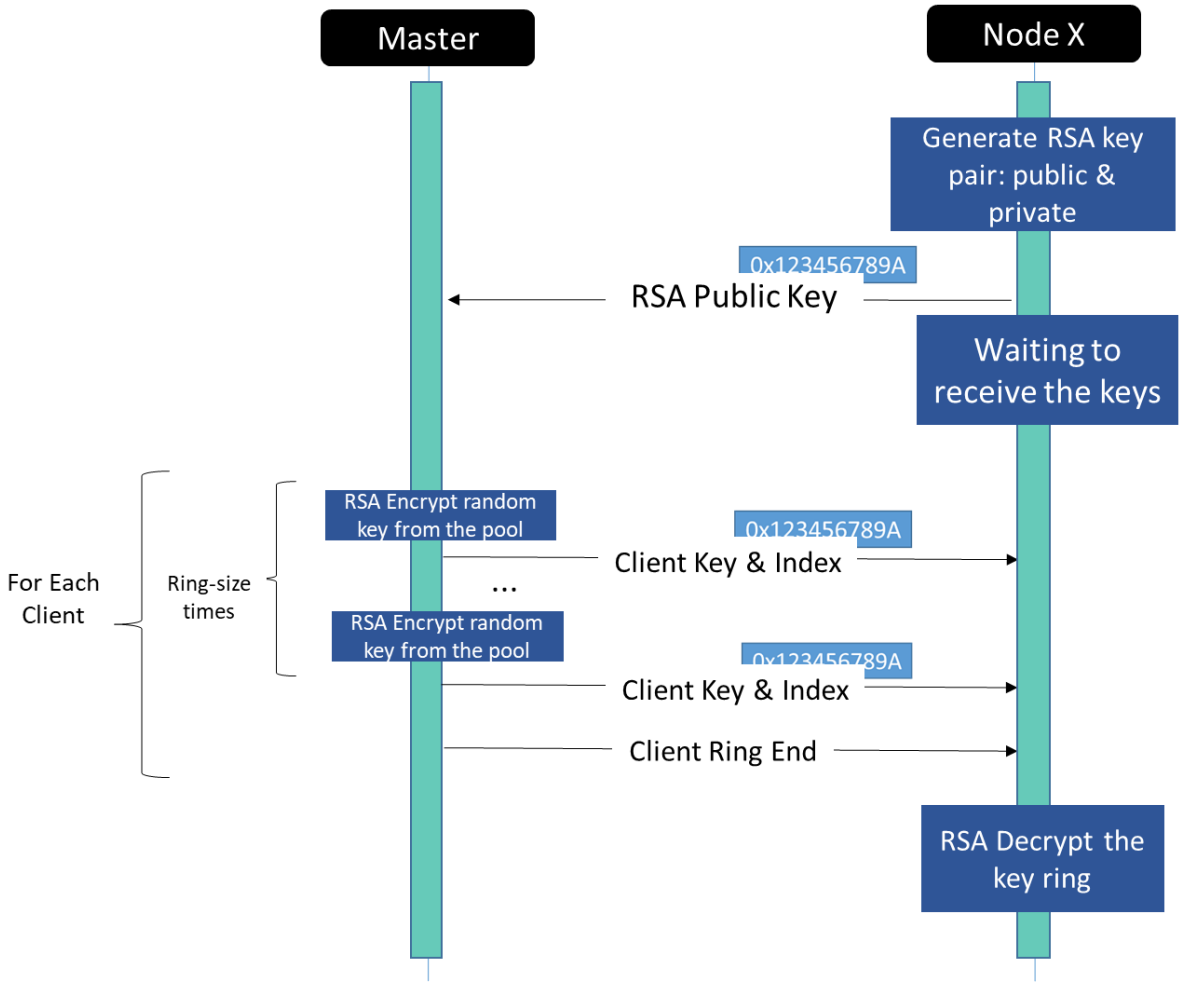
ה"מאסטר" מחשב את כמות המפתחות שהוא נדרש לייצר וכן את מספר המפתחות שעליו לחלק לכל צומת עפ"י נתונים פיזיים כמו גודל זיכרון וכן עפ"י ידע מוערך מוקדם כמו מספר הצמתים ברשת, ב'שכונה' וכ"ו. עפ"י חישובים מתמטיים הוא מגיע לגודל הבריכה אותו עליו לייצר, ומייצר אותם.

כל מפתח שהמאסטר מייצר נכנס למערך ששמור אצלו. אינדקס המפתח מוגדר כמיקומו במערך המפתחות. לאחר מכן המאסטר עובר למצב MASTER_DONE.



3. חלוקת מפתחות

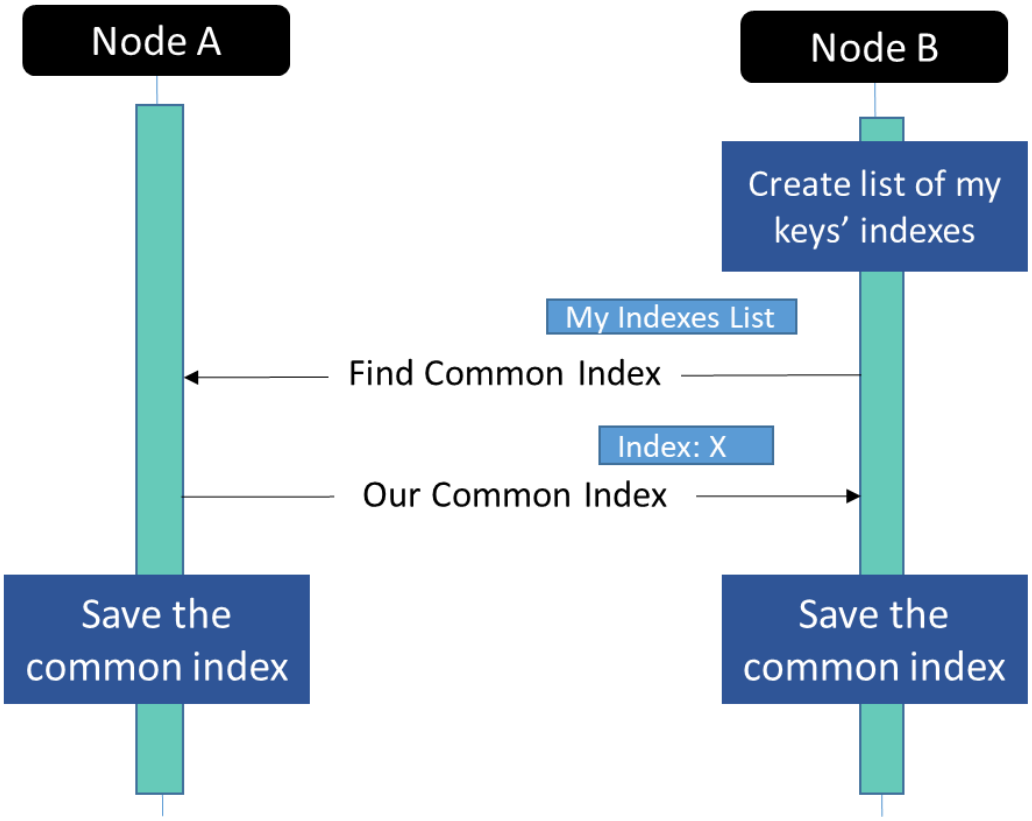
כל "לקוח" מייצר מפתח RSA פרטי ופומבי, ושולח את המפתח הפומבי ל"מאסטר" בהודעת CLIENT_PUBLIC_KEY על מנת שיוכל לשלוח לו את המפתחות מוצפנים. המאסטר מגריל k מפתחות מהמאגר ושולח ללקוח כל מפתח בהודעת CLIENT_RING_KEYS כשהוא מוצפן ע"י המפתח הפומבי של הלקוח ואליו מצורף אינדקס המפתח. בסיום שליחת כל המפתחות המאסטר שולח הודעת CLIENT_RING_END ואז הקליינט עובר למצב CLIENT_GOT_KEYS ומפענח את המפתחות שנשלחו בעזרת המפתח הפרטי שלו.



4. מציאת מפתח משותף

כל צומת שקיבל מפתחות שולח הודעת I_AM_ON_THE_NETWORK , על מנת ששאר הצמתים שנמצאים ברשת יכירו אותו.

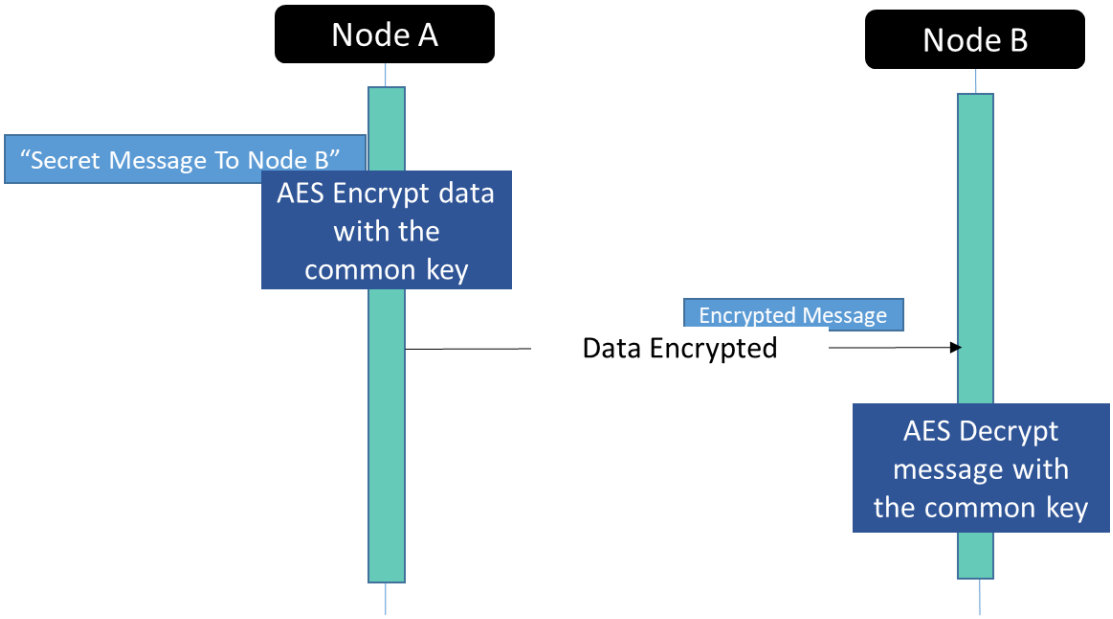
כאשר צומת רוצה לדבר עם צומת אחר ברשת, הוא שולח לו הודעת CLIENT_START_SESSION עם רשימת אינדקסי המפתחות שלו, הצומת שמקבל את ההודעה בודק חפיפה בין האינדקסים שקיבל לבין האינדקסים שלו, אם מצא מפתח משותף (יקרה בהסתברות של 80%-90%) הוא שולח לצומת הודעת CLIENT_COMMON_INDEX שבו המפתח המשותף, במקרה ולא קיים מפתח משותף, יוחזר 1-. שני הצמתים שומרים אצלם במערך השכנים את אינדקס המפתח איתו ידברו ביניהם.



5. רשת בטוחה

כאשר מכשיר רוצה לתקשר עם מכשיר אחר, הוא שולח הודעת MESSAGE_ENC_DATA שבו במידע שאותו רוצה להעביר מוצפן ע"י AES, כאשר מפתח ההצפנה הוא המפתח המשותף. המכשיר שמקבל את ההודעה, מפענח בעזרת אותו מפתח.

מכשיר חדש שנכנס לרשת, עובר את שלבי הפרוטוקול ובאפשרותו לתקשר בצורה מאובטחת עם כל מכשיר ברשת. וכן מכשיר שמתנתק מהרשת לא מפריע לתקשורת בין הצמתים האחרים, וכן לא חושף את כל מפתחות ההצפנה.



נוסחאות מתמטיות

לאחר שנבחר מאסטר ברשת, עליו לחשב את גודל בריכת המפתחות אותה עליו לייצר. חישוב זה אינו טריוויאלי כלל, ולוקח בחשבון גורמים רבים, כמו מס' הצמתים באופן כללי, גודל השכונה - מס' הצמתים ברשת (בטווח הקליטה אחד של השני), גודל הזיכרון הפיזי, גודל כל מפתח, מס' המפתחות שיחולקו לכל צומת, וכמובן ההסתברות הרצויה עבור מפתח משותף בין כל שני צמתים ברשת...

$$p' = 1 - \frac{(1-\frac{k}{P})^{2(P-k+1/2)}}{(1-\frac{2k}{P})^{2(P-2k+1/2)}}$$

פתירת הנוסחה המתמטית הזו נחשבת לבעיית NP קשה. כיוון שאין אפשרות לבודד את P (גודל הבריכה), אלא יש לבצע קירוב לפתרון ע"י מספר רב של איטרציות, על מספר התחלתי x0, אותו עלינו להגדיר.

על מנת לפצח את הבעיה נעזרנו במתמטיקאי שסייע להעביר את הנוסחאות לmatlab, ולאחר קבלת נוסחה עבור x0 כתבנו הקוד בפייתון.

הודעות

מבנה:

כל הודעה מכילה את הנתונים הבאים:

- סוג ההודעה: string המוגדר מראש המכיל את סוג ההודעה.
- מזהה המידע: מידע נוסף לגבי תוכן ההודעה. יכול להכיל אינדקס מפתח, iv של AES ועוד.
- מידע (data): תוכן ההודעה. יכול להיות מפתח RSA פומבי, מפתח AES מוצפן ועוד.

סידור לרצף בינארי:

על מנת לשלוח את ההודעה ברשת יש צורך לפרוס את מבנה ההודעה שהוא object להיות רצף תווים בינאריים. נשתמש בספריה pickle כדי לבצע זאת.

שליחה:

ההודעות ברשת נשלחות בפרוטוקול UDP שהוא פרוטוקול תקשורת מהיר אך ללא לחיצת יד ואימות של המידע הנשלח. השתמשנו ב UDP מכיוון שרצינו לממש את הפרוטוקול החדש בצורה קלה שמבצעת את התקשורת הנחוצה בלבד ללא עומס מיותר על הרשת ובאופן פשוט למכשירים. בUDP קיימת מגבלה על גודל החבילה הנשלח והוא בערך 556 בתים.

לכן היה צורך לחלק את המידע (הבינארי) בשליחת ההודעה לכמה שליחות. על מנת שהמקבל יידע כמה בתים אמורים להישלח בהודעה מסוימת הגדרנו שכל הודעה מתחילה ב4 בתים- "header" שמכילים את גודל ההודעה, ואח"כ תוכן ההודעה עצמה- "data". השולח מחלק את המידע הזה לבלוקים בגודל שנתמך בUDP ושולח. המקבל מקבל את 4 הבתים הראשונים- header, קורא את גודל ההודעה שאמורה להישלח ומחכה עד לקבלת מספר הבתים שצוינו בheader.

חוטים (threads)

בפרויקט השתמשנו ב-multi-threading:

א. עבור האזנה להודעות. כל צומת פותח thread נפרד להאזנה, על מנת שהאזנה להודעות תתאפשר תוך כדי הפעילות השוטפת.
ב. כאשר המאסטר שולח לצמתים ברשת את קבוצת תתי המפתחות שלהם, מוצפנים ב-RSA. מכיוון שהצפנת RSA לוקחת זמן, ובייחוד, כאשר עליו לשלוח כמות מפתחות עבור כל צומת, המאסטר יכול לאבד הודעות חשובות כמו הודעת IS_THERE_MASTER, ולגרום למופע נוסף של מאסטר. לא ניתן לפתור

זאת ע"י שליחה מהthread הראשי מכיוון שתוך כדי שליחה לצומת אחד יכולה להתחיל שליחה לצומת נוסף ואז נזדקק לthread-ים נוספים.

תיאור המערכת שמומשה

מימוש הפרוטוקול:

המבנה באופן כללי מוצג כך:

```
# ask "who is Master" in the network.
# return True if I am the Master, False otherwise
state.IAmMaster = client.find_master()
print("Master is found! master ip is: " + str(state.masterIP))

if(state.IAmMaster): # perform Master logic
        import master
        state.status = MASTER_INIT

        # Master calculations
        state.poolSize = master.calculate_pool_size()
        print("Calculated the pool size. The size is: " + str(state.poolSize))
        state.subKeysSize = master.calculate_sub_keys_size()
        print("Calculated the sub pool size. The size is: " + str(state.subKeysSize))

        # generate the keys
        state.pool_keys = master.generate_key_pool()
        for index, key in enumerate(state.pool_keys):
                state.keys.append((index,key))
        #print("my keys: "+str(state.keys))
        # send the sub-pool keys to nodes that send their public key
        while(state.toSendKeys != []):
                master.send_keys()
        state.status = MASTER_DONE
```

```
else: # perform client logic
        state.status = CLIENT_INIT
        #generate RSA key
        print("Generating RSA key...")
        state.RSAPublic, state.RSAPrivate = crypt.generate_asym_key()
        # send client public key to master
        messages.send_single_msg(state.masterIP, 'CLIENT_PUBLIC_KEY', 2048, state.RSAPublic)

        # wait until key are sent from Master
        print("Waiting to receive the keys from the Master...")
        while(state.status != CLIENT_GOT_KEYS):
                pass
        print('Recieved the keys from the master. Now decrypting... ')
        # decrypt all the keys
        state.keys = [(index, crypt.decrypt_asym(state.RSAPrivate, key)) for (index, key) in state.keys]
        print("Got "+ str(len(state.keys)) +" keys, My indexes are: "+ str([x[0] for x in state.keys]))
        # publish my IP in the network
        client.publishMe()
        state.status = CLIENT_DONE
```

1. אתחול ומציאת מאסטר

המערכת שלנו מורכבת מרשת קטנה של IoT שמחליטים על "מאסטר" באופן דינאמי.

```
def find_master():
        print("Looking for the Master on the network...")
        counter = 2 # two tries for looking after master, if there is a stack in the network
        while(state.status != MASTER_FOUND):
                print("Trying for the #" + str(3-counter) + " time...")
                messages.broadcast(messages.IS_THERE_MASTER) # ask if there is master on the network
                time.sleep(10) # wait for master response (it takes a long time for the IoT to respond... ~10 seconds!)
                counter -= 1
                if(counter == 0 and state.status != MASTER_FOUND): # after 2 tries to find the master, set myself as Master
                        messages.broadcast(messages.I_AM_MASTER)
                        state.masterIP = state.myIP
                        state.status =  MASTER_FOUND
                        print("No master is found! Setting myself as master")
                        return True
        return False
```

2. חישוב גודל מאגר המפתחות ויצירתו

המאסטר מחשב את גודל בריכת המפתחות שעליו לייצר, ומייצר אותם, מחשב את קבוצת תתי המפתחות שעליו לשלוח לכל מכשיר.

```
# x0, the first x, in order to begins the iterations of calculating p'
gP = -k**2/math.log(1-pp)

# function to calculate p'.
def func(P):
        return log(1-pp)-(2*P-2*k+1)*log(1-k/P)+(P-2*k+1/2)*log(1-2*k/P)

# function to calculate the size of the pool the master have to generate
def calculate_pool_size():
        P = int(fsolve(func, gP)[0])
        return P
```

3. שליחת מפתחות באופן מוצפן

הלקוח מייצר מפתח RSA פומבי ופרטי (הלקוח שולח למאסטר את החלק הפומבי).

```
def generate_asym_key(bits=2048):
        # default exponent is: 65537 (0x10001)
        new_key = RSA.generate(bits)
        public_key = new_key.publickey().exportKey("PEM")
        private_key = new_key.exportKey("PEM")
        return public_key, private_key
```

המאסטר שולח את המפתחות מוצפנים

```
# function that sendings keys to clients that waiting for them, encypted with thier public key
def send_keys_to_client(ip, clientPublicKey):
        print("Sending keys to: " + str(ip) + "...")
        # Send sub-pool of size determined in calculate_sub_keys_size()
        for i in range(state.subKeysSize):
                # Get random key from the pool
                key_index = int(math.floor(random.random() * len(state.pool_keys)))
                keyData = state.pool_keys[key_index]
                # Encrypt the key with the client's public RSA key
                keyData = crypt.encrypt_asym(clientPublicKey, keyData)
                # Send to client
                messages.send_single_msg(ip, 'CLIENT_RING_KEYS', key_index, keyData)
        # Indicate the client that no more keys will be sent
        messages.send_single_msg(ip, 'CLIENT_RING_END')
        print("Finish to send keys to client.")
```

```
def encrypt_asym(pub_key, message):
        key = RSA.import_key(pub_key)
        cipher = PKCS1_v1_5.new(key)
        ciphertext = cipher.encrypt(message)
        return ciphertext
```

הלקוח מקבל קבוצה של תתי מפתחות מוצפנים, ומפענח ע"י המפתח הפרטי שיצר.

```
def decrypt_asym(priv_key, ciphertext):
        key = RSA.import_key(priv_key)
        cipher = PKCS1_v1_5.new(key)
        message = cipher.decrypt(ciphertext, None)
        return message
```

4. מציאת מפתח משותף

צומת שולח לצומת אחר את אינדקסי המפתחות שלו.

הצומת המקבל מבצע חיתוך עם האינדקסים שלו, ושולח חזרה את אינדקס המפתח המשותף.

```
common_key = -1
# Intersect my keys & the sender keys
common_keys = list(set(message.data).intersection([x[0] for x in state.keys]))
# If common key is found- update internal state
if common_keys:
        for index, neighbor in enumerate(state.neighbors):
                list_neighbor = list(neighbor)
                if list_neighbor[0] == ip:
                        list_neighbor[1]= common_keys[0]
                state.neighbors[index] = tuple(list_neighbor)
        common_key = common_keys[0]
# Response with the common key index, or -1 if not found
send_single_msg(ip, CLIENT_COMMON_INDEX,common_key)
```

שני הצמתים שומרים את האינדקס המשותף.

```
for index, neighbor in enumerate(state.neighbors):
        list_neighbor = list(neighbor)
        #print("list_neighbor: "+str(list_neighbor))
        if list_neighbor[0] == ip:
                list_neighbor[1] = message.dataID
        state.neighbors[index] = tuple(list_neighbor)
```

5. רשת בטוחה

כאשר שני צמתים מתקשרים ביניהם הם משתמשים במפתח המשותף כדי להצפין ולפענח הודעות.

```
# Find the common index of me and the neighbor
key = ''.join([key for (i, key) in state.keys if i == index])
msg = "This is a very secret message from: "+ state.myIP
print("The message is: " + str(msg))
# Encrypt the message with AES (key is 256 bit)
iv, cipher = crypt.encrypt_message(key, msg)
print("The cipher is: "+ str(cipher))
messages.send_single_msg(neighbor, messages.MESSAGE_ENC_DATA, iv, cipher)
```

```
for neighbor, index in state.neighbors: # look for the sender ip in the neighbors list
        if neighbor == ip: # neighbor is found
                # supposed to find only one key with the same index!
                key = ''.join([key for (i, key) in state.keys if i == index])
                msg = crypt.decrypt_message(key, message.data, message.dataID)
                print("Decrypted message from: "+ str(ip) + ". Message is: " + str(msg))
```

נוסחאות מתמטיות

נעזרנו בספריות numpy, scipy על מנת לפתור את הנוסחה.

```
# x0, the first x, in order to begins the iterations of calculating p'
gP = -k**2/math.log(1-pp)

# function to calculate p'.
def func(P):
        return log(1-pp)-(2*P-2*k+1)*log(1-k/P)+(P-2*k+1/2)*log(1-2*k/P)

# function to calculate the size of the pool the master have to generate
def calculate_pool_size():
        P = int(fsolve(func, gP)[0])
        return P
```

מימוש שליחת/ קבלת הודעות

מבנה ההודעה:

```
# the messages have three fields:
# 1. type = the header of the message
# 2. dataID = an aditional data, like the index of key in CLIENT_RING_KEYS message , when no nedded is 0
# 3. data = the data itself. when the message don't have data, it'w None
class Message(object):
        def __init__(self, type, dataID, data):
                self.type = type
                self.dataID = dataID
                self.data = data
```

סידור לרצף בינארי בשליחה:

```
def send_single_msg(ip, type, dataID=0, data=None):
        print("Sending message " +str(type)+" to: "+ str(ip))
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        if ip == '<broadcast>':
                s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
                s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
        # serialize the message to a byte stream
        msg = Message(type,dataID,data)
        bits = pickle.dumps(msg) # format the msg
        send_msg(s, bits, ip)
        s.close()
```

פריסה מחדש לתוך class Message:

```
# listen for message, if it is not sent by me - process the message
def listen(socket):
        while True:
                bits , address = get_msg(socket)
                if my_ip != address[0]:
                        break
        msg = pickle.loads(bits) # Returns the message to a format of Message class
        return msg, address[0]
```

שליחת ההודעה:

```
# send block of msg
def _send_block(s, data, ip):
        while data:
                data = data[s.sendto(data, (ip,PORT)):]

# send all the message
def send_msg(s, data, ip):
        header = struct.pack('>i', len(data))
        _send_block(s, header, ip)
        _send_block(s, data, ip)
```

קבלת ההודעה:

```
# get block of msg
def _get_block(s, count):
        if count <= 0:
                return ''
        buf = ''
        while len(buf) < count:
                buf2, address = s.recvfrom(count - len(buf))
                if not buf2:
                        # error or just end of connection?
                        if buf:
                                raise RuntimeError("underflow")
                        else:
                                return ''
                buf += buf2
        return buf, address


# get all the message
def get_msg(s):
        header, ip = _get_block(s, 4)
        count = struct.unpack('>i', header)[0]
        return _get_block(s, count)
```

חוטים (threads)

פתיחת thread נפרד עבור האזנה להודעות ברשת:

```
def async_listen_to_messages():
        print("Creating New thread..")
        c = threading.Thread(target=async_listen)
        c.daemon = True
        c.start()

def async_listen():
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.bind(('',PORT))
        print("Created socket. Listening for messages...")
        while(True):  # may stop the thread on some condition...
                msg, ip = listen(s)  # block until message accepted
                process_message(msg, ip)
```

פתיחת thread עבור שליחת המפתחות מוצפנים ב-RSA:

```
elif message.type == CLIENT_PUBLIC_KEY: # only the master can get this msg
        if state.status == MASTER_INIT:
                state.toSendKeys.append((ip,message.data)) # save the client ip and public key for later use
        elif state.status == MASTER_DONE:
                # encrypt the sub-pool with the client's public key
                c = threading.Thread(target=send_keys_to_client, args=[ip, message.data])
                c.daemon = True
                c.start()
```

מכשירים

במהלך הפרויקט ניסינו לעבוד עם כמה מכשירים מסוגים שונים כמו:

- LinkitSmart
- esp32
- windows pc
- Raspberry Pi 3

חלק ממכשירים אלו היו IoT מהסוג הפשוט ביותר, עם מעט מאד זיכרון, מעבד חלש מאד, ובעיקר הכילו micro-python בלבד. דבר זה לא אפשר לנו להתקין ספריות כבדות כמו numpy, scipy, pyCryptodome.

לאחר מאמצים רבים, עבודה עם סביבות עבודה שונות ומגוון מערכות הפעלה, הגענו למסקנה שאנחנו צריכות לשנות גישה וללכת על מכשירים קצת יותר חזקים שמכילים פייתון בגרסתו המלאה, והחלטנו לעבור להשתמש ב Raspberry Pi 3.

בדיקות - תכנון וביצוע

בדיקות מקיפות של כל פונקציות המערכת ע"מ לוודא נכונות, מקרי קצה, מקרים חריגים ועוד.

בדיקות מתמטיות

בקובץ test_pool.py הרצנו בדיקות מקיפות על המתמטיקה וההסתברויות, חיתוכים, נכונות הנוסחאות וכו'

- מציאת C אופטימאלי שייתן תוצאות כמה שיותר גבוהות להסתברות לחפיפה.
- בדיקה שהמסטר יכול לייצר בריכת מפתחות בגודל P.
- הגרלת תתי קבוצות של מפתחות בגודל k, לפי כמות הצמתים ברשת. בדיקה שקיים חיתוך בין כל שני תתי קבוצות שהוגרלו, בהסתברות של 80%-90%.

בדיקות קריפטוגרפיות

בדיקות שפונקציות ההצפנה עובדות כנדרש, מצפינות ומפענחות את המידע כצפוי. בקובץ test_crypt.py נמצאות כל הבדיקות שנערכו.

- נוצר מפתח RSA חוקי
- מידע שמוצפן בRSA אינו גלוי (cipher). פיענוח בRSA מחזיר את המידע המקורי
- מידע שמוצפן בAES אינו גלוי (cipher). פיענוח בAES מחזיר את המידע המקורי

בדיקות תקשורת

התבצעו בבדיקות יחידה

- כל סוג וגודל של הודעה בפרוטוקול עובר בשלמות מצומת לצומת ומטופל בהתאם לפרוטוקול
- המכשיר הראשון שנכנס לרשת נהיה מאסטר ומיידע על כך את כל הצמתים הבאים אחריו
- בדיקה שהמאסטר הוא יחיד ברשת
- בכל שלב, כל צומת שנכנס לרשת מקבל תת קבוצת מפתחות.
- לאחר שצומת עוזב את הרשת, שני צמתים אחרים יכולים להמשיך לתקשר ביניהם בצורה מאובטחת.
- צומת שעזב יכול לחזור בכל עת, ויקבל קבוצת מפתחות חדשה.

בדיקות תאימות

- בדיקה שהתהליך עובד על מכשירי IoT בעלי חומרה/תוכנה שונה. כרגע בדקנו על מכשירי Raspberry Pi ו-Windows PC (בעלות מערכות הפעלה שונות). לכאורה התהליך אמור לעבוד על כל התקן בעל python (ולא micro-python).

בדיקות תחזוקה

- הקוד כתוב בצורה פשוטה, ברורה ומתועדת.

מסקנות

- גילינו שבפרויקט מחקרי מסוג זה שאנו עושות, עיקר העבודה אינה מתבטאת בכמות הקוד ומימוש נרחב של תוכנה וכן אף לא על תוצר מהפכני בהיקפו, אלא בהרחבת הידע, בהכרת עולם המושגים של נושא טכנולוגי לעומקו ובהבנה של תהליכים חדשים ומורכבים ויישומם בפועל.
- שימת דגש על מחשבה ותכנון לפני כל כתיבת קוד בפועל, העלאת ההחלטות על הכתב וניסוחן מקלה על המימוש, מונעת בלבול ומאפשרת התמקדות בפתרון הבעיה.
- עבודה עם טכנולוגיות חדשות ולא מוכרות כמו התקני הIoT שהתעסקנו איתם דורשת חיפוש יסודי בכל מאגרי המידע הנגישים תוך פניה למספר כיוונים והצלבת פיסות מידע באופן מושכל. לאחר מכן נצרך ביצוע החלטה "עיוורת" במובן מסוים כדי להשיג התקדמות כלשהי, מכיוון שבשלב מוקדם כזה של התפתחות מוצר חדש ועתידי לא קיים "מתכון" בטוח שיביא לנו את התוצאות הרצויות.

סקירת עבודות דומות בספרות והשוואה

סקרנו מס' פתרונות אפשריים שהוצעו עבור אבטחה ב- IoT, אולם לכל אחד מן הפתרונות קיימים חסרונות:

- שימוש בענן

ישנו פתרון אבטחה להתקני IoT המסתמך על שימוש בענן על מנת להשיג את האבטחה הרצויה. כל התקן ברשת יתחבר לשירות בענן שיתמוך בניהול האבטחה של כל ההתקנים, ומולו יתבצעו פעולות האימות וההצפנה[6].

פתרון זה דורש ניהול של שירות בענן, וזו תקורה שדורשת משאבים נוספים, בנוסף, יהיה צורך בהגדרת ספקים אמינים לכזה שירות, ופתרון במקרה של התחזות. במחקר שלנו אנו מנסים להביא את השליטה באבטחה להיות בבלעדיות אצל רשת ההתקנים המקומית, ללא קישור לשירות חיצוני.

- שימוש בTPM:

פתרון זה מסתמך על מכשיר הTPM לביצוע פעולות קריפטוגרפיות ונתינת שירותי הצפנה למכשירים ברשת[7]. זה נראה אמנם רעיון ישים, אך החיסרון בו שדרוש מכשיר TPM כזה עבור כל רשת IoT, וזה גובה עלות נוספת. בפרויקט שלנו אנו ממשים פתרון אבטחה שמתבסס אך ורק על מכשירים הקיימים ברשת, ללא עלות נוספת.

- ארכיטקטורת IoT מאובטחת לערים חכמות המטפלת בפגיעויות במערכות IoT:

הארכיטקטורה כוללת רשתות שחורות ומערכת ניהול מרכזית (KMS) המספקות סודיות, שלמות, פרטיות והפצה מרכזית יעילה. המטרה הייתה לספק שירותי אבטחה הממתנים את הפגיעות של רשתות IoT בשכבות הקישור והרשת, במיוחד עבור נתונים קריטיים.
החסרונות של גישה זו כוללים היעדר פתרון פרטיות להגדרת מיקום המכשיר ואיתור ניתוב חדש עבור צמתים, מה שמוביל לאובדן נתונים.

- ארכיטקטורת SDN לפיתוח יישומי IoT:

ארכיטקטורת SDN אומצה על מנת לספק בסיס לפיתוח מערכת מאובטחת שמאפשרת למנהלי מערכות להציג את העולם באופן גלוי של איומים אפשריים להתקפות ברשת ה- IoT ולספק להם את הזכות לשלוט ברשת מפני האיומים.
עם זאת, אבטחה, מדרגיות ואמינות הן חלק מהחסרונות של רשתות SDN. ההפרדה בין מטוסי הבקרה והנתונים של ה- SDN גורמת לביצועים ירודים בעיבוד חבילות, אשר מובילה לבעיות משמעותיות, כגון עיכוב או אובדן של חבילות והתקפות DoS (DDoS).

- ארכיטקטורת אבטחה חדשה המבוססת על SDN עבור ה- IoT, הידועה גם בשם תחום ה- SDN באמצעות בקרי הגבול:

המחברים תיארו כיצד ניתן להשתמש ב- SDN כדי לחבר בין התקני IoT הטרוגניים, כיצד ניתן לשפר את האבטחה של כל דומיין, וכיצד ניתן לחלק את כללי האבטחה מבלי לפגוע בביטחון של כל תחום.
עם זאת, המחברים לא היו מסוגלים להתמודד עם האתגר של הבטחת תעבורה רצויה ולא רצויה והגנה על הארגון, אשר הם החסרונות העיקריים של שימוש בבקרי הגבול.

- פרוטוקול לניהול מפתחות בצורה קלה:

הפרוטוקול תלוי בהתאמות של רכיבי אבטחה שונים ב- IoT כדי להגדיר ערוצי תקשורת מאובטחים ומוגנים עבור IoT. במהלך העברת הנתונים לאורך הערוץ, הפרוטוקול מבטיח סודיות נתונים ואימות צומת מוגבל.
עם זאת, פרוטוקול האבטחה מוגבל, ואינו מפרט את ההתאמה הנדרשת בין תקורה לתקשורת לבין מספר צדדים שלישיים[8].

מערכות ניהול הפרויקט:

#	מערכת	מיקום
1	מאגר קוד	https://github.com/reutnagar/distributed-RSA-for-IoT
2	יומן	https://trello.com/b/DkjV5sEx/a
3	סרטון שלב אלפא	https://drive.google.com/file/d/1yG61mZb-n9U0TWnGObKPunGcjKk5-YSS/view
	סרטון סופי	

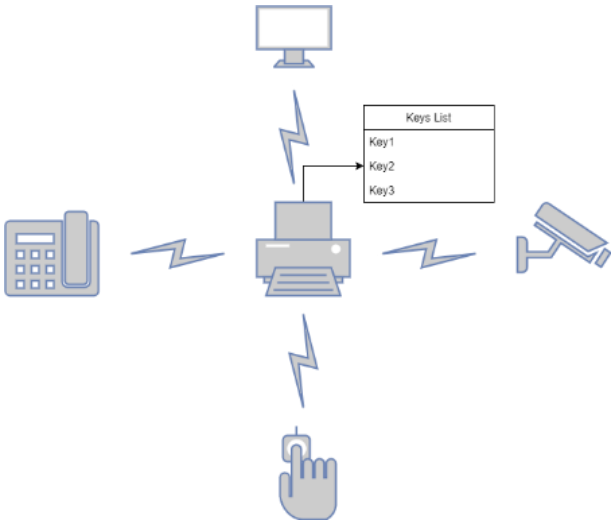
נספחים

א. רשימת ספרות \ ביבליוגרפיה

[1] "אבטחת מידע – ויקיפדיה." [Online]. Available: https://he.wikipedia.org/wiki/אבטחת_מידע. [Accessed: 19-Nov-2017].
[2] "RSA – ויקיפדיה." [Online]. Available: https://he.wikipedia.org/wiki/RSA. [Accessed: 19-Nov-2017].
[3] Wikipedia. (n.d.). *AES Wikipedia*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[4] "האינטרנט של הדברים – ויקיפדיה." [Online]. Available: https://he.wikipedia.org/wiki/האינטרנט_של_הדברים. [Accessed: 19-Nov-2017].
[5] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," *Proc. 9th ACM Conf. Comput. Commun. Secur.*, pp. 41–47, 2002.
[6] M. Tao, J. Zuo, Z. Liu, A. Castiglione, and F. Palmieri, "Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 1040–1051, 2018.
[7] H. Hamadeh, S. Chaudhuri, and A. Tyagi, "Area, energy, and time assessment for a distributed TPM for distributed trust in IoT clusters," *Integr. VLSI J.*, vol. 58, no. December 2016, pp. 267–273, 2017.
[8] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *J. Netw. Comput. Appl.*, vol. 88, no. March, pp. 10–28, 2017.

ב. תרשימים וטבלאות

מראה רשת בעלת RSA מבוזר ב"בית חכם". בדוגמה זו המכשיר המנהיג הוא המדפסת. בידיו נמצאת רשימת המפתחות, אותה הוא יחלק לשאר המכשירים בבית באמצעות תקשורת אלחוטית.



Abstract

Over the past few years, IoT has been gaining momentum. These devices become the target of attacks. Because their resources are poor, existing security solutions cannot be implemented on them and therefore lack adequate security.
This project deals with finding a security solution for small IoT networks. The solution is also possible in large networks implemented hierarchies.
We present a new security protocol consisting of five steps:

- Finding a Master
- Finding its size and creation of the key pool

- Distribution of keys
- Finding a common key
- Secure network

For the second step of the protocol we relied on a mathematical paper that measured the size of the key pool to be created in order to overlap at least one key between two nodes in the network with high probability. This is in order for the network to be dynamic, and changes to the network topology will not undermine its security.
We implemented the solution on three "Raspberry Pi 3" IoT devices, one of which is the master and the other two are two nodes on the network.

Department of Software Engineering

Final Project - 2018

Probability Based Keys Sharing for IOT

Azrieli
College of Engineering
Jerusalem

עזריאלי
מכללה אקדמית להנדסה
ירושלים

By

Sarie Safrin 312548779

Reut Nagar 205437874

Academic Host: Dr. Guy Leshem Date:

Project Supervisor: Assaf Spanier Date:

Ramat Bet Hakerem, P.O.B 3566, Jerusalem 9103501 רמת בית הכרם, ת.ד. 3566, ירושלים מיקוד
Tel. +972-2-6588000 | Fax. +972-2-6588001 | Email: info@jce.ac.il | Web: www.jce.ac.il