הצפנה מקצה לקצה - המשך

מאת עידן שכטר

הקדמה

במאמר הקודם הבנו את החשיבות של הצפנה מקצה לקצה בימינו, ואף למדנו כיצד לממש מנגנון בסיסי בעצמנו, כזה שעל פניו נראה נטול בעיות ופגמים.

במאמר זה נבצע עליית מדרגה ונבין את הקושי במימוש מנגנון הצפנה מקצה לקצה כאשר איום הייחוס גבוהה, לדוגמא, במצב בו המפתח הפרטי של אחד הצדדים נגנב.

מצב זה אמנם נשמע חריג (והוא אכן כך, מאחר והאדם הפשוט לא נמצא תחת איום ייחוס שכזה), אך מימוש פרוטוקולי הצפנה מורכבים ומתקדמים הפך להיות סטנדרט - אפליקציות פופולריות רבות עובדות קשה כדי לאפשר גם ל"משתמש הפשוט" להנות ממנגנוני אבטחה המשפרים את הפרטיות שלו פלאים.

התרחיש שנרצה למנוע

כפי שלמדנו במאמר הקודם, הסוד המשותף המיוצר על ידי שני משתמשים המעוניינים לתקשר בצורה מוצפנת הוא "נגזרת" של מפתחות ההצפנה שלהם, או בצורה קצת יותר מדויקת - של המפתחות הפרטיים שלהם.
כמובן שגם למפתחות הפומביים יש חשיבות, והם נחוצים לטובת חישוב הסוד המשותף, אבל אותם ניתן להשיג בקלות לעומת המפתחות הפרטיים, עליהם נשאף לשמור ככל שביכולתנו.

התרחיש עליו נדבר במאמר זה מתחיל כאשר המפתח הפרטי של אחד הצדדים מגיע לגורם שלישי ללא ידיעתו, מצב המאפשר לאותו גורם לייצר את הסוד המשותף ולפענח את ההודעות המוצפנות הנשלחות בין שני המשתמשים ו"לשבור" את מנגנון ההצפנה מקצה לקצה.

מקור הבעיה

הפתרון שנציע לבעיה מתבסס על העובדה שגורם שלישי בהכרח הצליח לשים ידו על המפתח הפרטי שלנו. בהנחה והמפתח הפרטי שלנו נמצא בידיו של גורם שלישי שכעת מסוגל לפענח הודעות מוצפנות, למה שלא פשוט נחליף את המפתח הפרטי שלנו אחת לכמה זמן? אם המפתח אכן נגנב, בשלב מסוים הוא יוחלף במפתח חדש ויהפוך לא רלוונטי!

בנוסף, בהנחה ונדאג להיפטר ממפתחות ישנים, פתרון זה מונע מגורם שלישי שהשיג את המפתח הפרטי הנוכחי לפענח הודעות שנשלחו על סמך מפתחות ישנים יותר (מאפיין המכונה "סודיות מושלמת קדימה", בו נגע בהמשך המאמר).

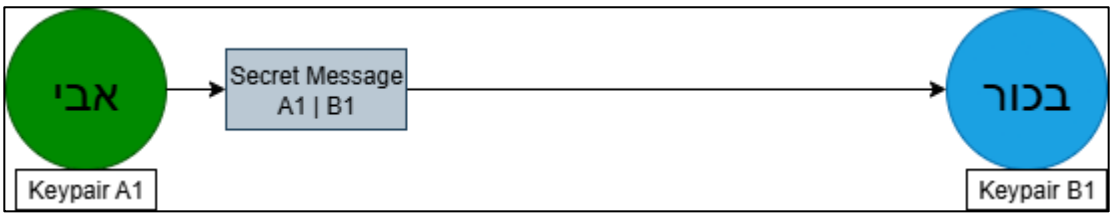
אבל רגע, אם הגורם השלישי כבר הצליח להשיג את המפתח הפרטי שלנו, למה שלא יצליח לעשות זאת שוב ולהשיג גם את המפתח החדש?

גם אם נתעלם מעובדה זו, ניסיון מימוש מנגנון שכזה ילמד אותנו שניהול שיחה מוצפנת מתמשכת תוך כדי חידוש מפתחות היא אינה פעולה של מה בכך.

ריענון מפתחות

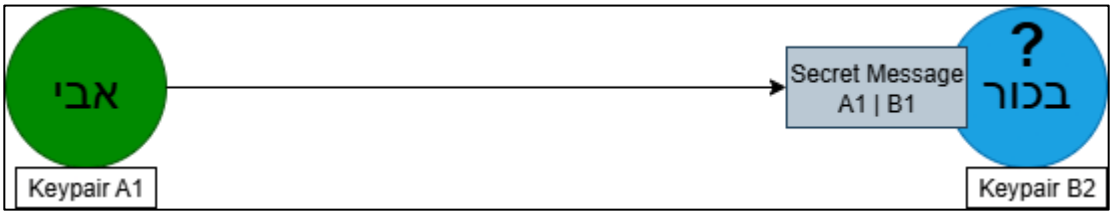
הבעיה המרכזית במימוש מנגנון שמבצע "ריענון" מחזורי של מפתחות הצפנה היא הקושי בסנכרון בין שני הצדדים לאורך זמן.

כאשר אבי שולח לבכור הודעה, ההודעה תוצפן באמצעות סוד משותף המבוסס על המפתחות האסימטריים שיש לכל אחד מהם באותו רגע נתון:

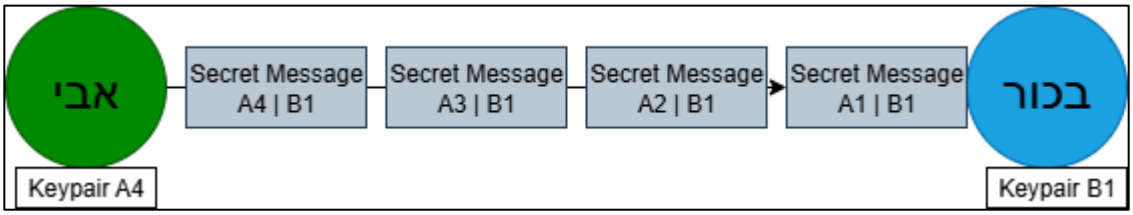


דוגמא פשוטה לבעיית סנכרון היא מצב בו אבי ישלח לבכור הודעה על סמך צמד המפתחות B1 (כלומר, ישתמש במפתח הפומבי B1 על מנת לחשב את הסוד המשותף) אך בדיוק ברגע שיסיים להצפין את ההודעה וישלח אותה לשירות, בכור יבצע ריענון מפתחות ויעבור להשתמש בצמד המפתחות B2.

מאחר ובכור לא שומר את המפתחות הישנים שלו, לא יוכל לפענח את ההודעה שקיבל מאבי. (נוכל לפתור את בעיה זו בכך שנוסיף לפרוטוקול מנגנון ש"נועל" מפתחות בין שני הצדדים, ומוודא שהמפתח בו נעשה שימוש ברגע הנתון אכן עדיין רלוונטי, אך פתרון זה לא פותר בעיות משמעויות אחרות).



דוגמא נוספת לבעיית סנכרון היא מצב בו אחד הצדדים לא מתחבר לשירות הרלוונטי במשך זמן, ובכך לא מעדכן את המפתחות שלו, בעוד שהצד השני ממשיך לשלוח לו הודעות ולהחליף מפתחות תוך כדי.



במידה והמפתח של בכור הוא זה שנגנב, פענוח ההודעות עדיין יהיה אפשרי על ידי גורם שלישי, וזאת כי רק הצד השני החליף את המפתחות שלו (חישוב הסוד המשותף יתבסס על המפתח הפרטי של בכור, זה שנגנב, ועל המפתח הפומבי הנוכחי של אבי, שניתן להשיג בקלות).

בנוסף, בכור לא יוכל לפענח הודעות שהוצפנו על בסיס המפתחות A1, A2, A3 של אבי, וזאת משום שאבי רענן את המפתחות שלו מספר פעמים בזמן שבכור לא היה מחובר לשירות.

One Keypair to Rule Them All

בעיה משמעותית נוספת שלאו דווקא נובעת משימוש במפתחות סטטיים (חידוש מפתחות לא ימנע את הבעיה) היא העובדה שזוג מפתחות בודד ישמש את המשתמש כדי להצפין הודעות בכלל השיחות של השירות, לכן גורם שלישי השם ידו על המפתח הפרטי יוכל לחשב את הסוד המשותף עבור כל שיחה רלוונטית, ולראות את כלל ההודעות של המשתמש.

אנו למדים ששימוש במנגנון הצפנה שתיארנו, כזה הנחשב לפרימיטיבי ועושה שימוש במפתחות קבועים באופן רחב, אינו פתרון יעיל ומספק כאשר מדובר באיום הייחוס הרלוונטי.

לשם כך, נדרש פתרון חדשני יותר. אפליקציית סיגנל, הנחשבת לאחת מאפליקציות המסרים המיידיים המאובטחות ביותר כיום, מציעה פתרון יפיפה לבעיה זו בפרוטוקול קוד פתוח המכונה גם הוא סיגנל, פרוטוקול בו עושות כיום שימוש אפליקציות כמו וואטסאפ, פייסבוק וסקייפ.

אפליקציית Signal

סיגנל התחילה את דרכה ב-2010 כאפליקציה בשם TextSecure, אשר אפשרה למשתמשיה לתקשר בצורה מוצפנת מקצה-לקצה בקלות, כחלק ממיזם בשם Whisper Systems שהוקם על ידי החוקר והיזם מוקסי מרלינספייק.

לאחר ש-Whisper Systems נרכשה על ידי Twitter (כיום X), אפליקציית TextSecure הופצה חינם כקוד פתוח.

זמן לא רב לאחר מכן, מוקסי עזב את Twitter והקים את Open Whisper Systems במטרה להמשיך ולפתח את TextSecure ואפליקציה נוספת, RedPhone, אשר אפשרה למשתמשיה לנהל שיחות קוליות מוצפנות מקצה לקצה.

ב-2015 Open Whisper System מיזגו את האפליקציות TextSecure ו-RedPhone אל אפליקציה יחידה בשם Signal, שהפכה עם השנים ל-Signal שאנו מכירים היום.

בימים אלה, אפליקציית Signal מפותחת על ידי Signal Messenger LLC, אשר ממומנת על ידי Signal Technology Foundation - ארגון ללא מטרות רווח שהוקם על ידי מוקסי מרלינספייק ובריאן אקטון (ממקימי WhatsApp).

השפעה

TextSecure הייתה מהאפליקציות הראשונות שהציעו הצפנה מקצה לקצה והובילה תהליך משמעותי שהפך את ההצפנה החזקה למיינסטרים - שינוי תפיסה עצום לעומת העבר, שבו פרטיות נחשבה כמשהו עבור "אלו שיש להם מה להסתיר".

> ***"Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say"***
>
> ***- Edward Snowden***

מעבר לכך שאפליקציית סיגנל היא חינמית וקוד פתוח, גם השרת של סיגנל מופץ כקוד פתוח, דבר המאפשר לאזרחים במדינות בהן הגישה לשירותי סיגנל חסומה להרים תשתית מקומית משלהם, לתקשר בצורה בטוחה ולהבטיח את החירות שלהם לפרטיות.

פרטיות הינה אידיאולוגיה עבור סיגנל, ומעבר לתחזוקה של פרוטוקול הצפנה מדהים כקוד פתוח, סיגנל נמנעת באופן מוחלט משמירה של מידע על המשתמשים שלה, ולכן גם אם תשתף פעולה עם רשויות, תוכל לספק להם מידע מינימלי. עובדה זו הופכת את השימוש באפליקציה לאטרקטיבי עוד יותר עבור אותם אינדיבידואלים המעוניינים להבטיח את הפרטיות שלהם. (כמו כן, אי אפשר להתעלם מהעובדה שמדובר בחרב פיפיות.

בדיוק כמו קריפטו ו-TOR, גם היתרונות של סיגנל מנוצלים כדי לברוח מהחוק ולבצע פשעים).

פרוטוקול Signal

מאחר ומדובר בפרוטוקול מסובך ורחב, לא נצלול אל עומק כל פרטיו הטכניים.

המרכיבים העיקריים עליהם נדבר הם פרוטוקול החלפת המפתחות X3DH ואלגוריתם Double Ratchet, עליהם ניתן להסתכל כמנגנון דו שלבי המאפשר לשני משתמשים לתקשר בצורה מוצפנת תוך כדי חידוש מפתחות מתמשך, גם כאשר אחד הצדדים אינו מחובר!

X3DH

פרוטוקול X3DH (Extended Triple Diffie-Hellman) מאפשר לשני משתמשים לייצר סשן באמצעותו יוכלו לתקשר באופן מוצפן ובטוח.

על סשן ניתן להסתכל כמצב קריפטוגרפי משותף בין שני משתמשים, המכיל את המידע הדרוש להם כדי לתקשר על פי הפרוטוקול הרלוונטי.

לדוגמא, ע"פ הפתרון הפרימיטיבי עליו דיברנו עד כה, יצירת הסשן שקולה ליצירת סוד משותף בעזרת DH. במקרה של סיגנל, המשתמשים יעשו שימוש ב-X3DH, גרסה "משודרגת" של DH, על מנת לייצר סוד משותף שבעזרתו יוכלו להתחיל לתקשר האחד עם השני.

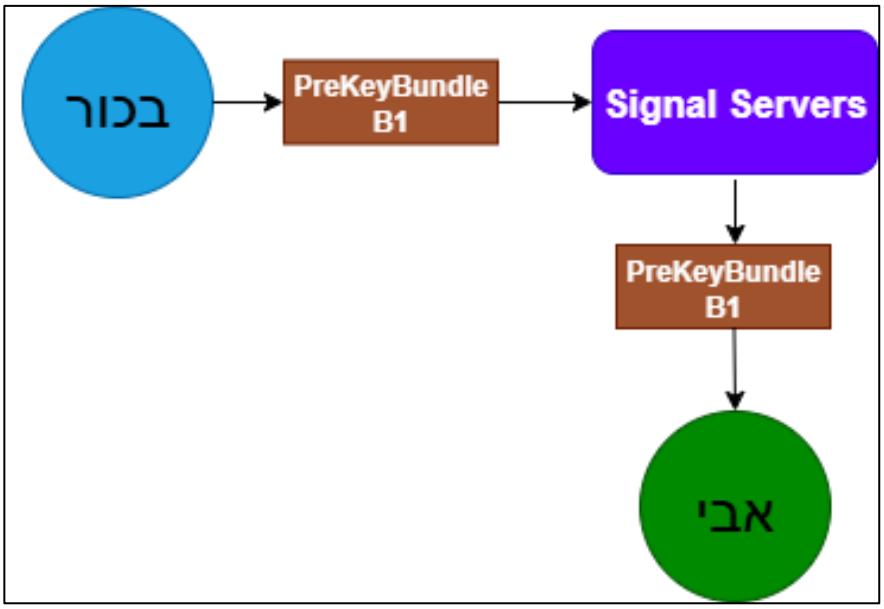
על פי פרוטוקול סיגנל, כל משתמש מחזיק זוג מפתחות אסימטריים הנקראים **Identity Keys**, אשר משמשים כזהות הקריפטוגרפית שלו. בשונה מהמקרה הקלאסי עליו דיברנו עד כה, זוג המפתחות משמש לאותנטיקציה ולא להצפנה.

מפתחות ההצפנה בסיגנל מבוססים עקומים אליפטיים (EC) ומשמשים לביצוע Diffie-Hellman וגם לחתימה דיגיטלית, לדוגמא 25519Ed.

מאחר ולא נצלול לחתימות דיגיטליות במאמר זה, עיקרון שחשוב לזכור הוא שחתימה דיגיטלית אשר בוצעה על ידי מפתח פרטי, תוכל להיות מאומתת על ידי המפתח הפומבי המתאים. דבר המאפשר למשתמש להוכיח שהוא מחזיק במפתח פרטי מסוים ובכך להוכיח את הזהות שלו.

Key Publishing

כל משתמש בסיגנל מעלה לשרתים של סיגנל אחת לכמה זמן "חבילת" מפתחות הנקראת PreKeyBundle המאפשרת למשתמשים אחרים לייצר מולו סשן קריפטוגרפי, גם כאשר הוא אינו מחובר לשירות! כאשר אבי ירצה לייצר סשן מול בכור כדי לשלוח לו הודעה, הוא יפנה לשרתי סיגנל ויבקש לקבל PreKeyBundle רלוונטי.



PreKeyBundle מכיר מספר פרמטרים שונים, אך אלה הנחוצים לביצוע X3DH הם:

Public Identity Key - מפתח הזהות הפומבי של בכור, מפתח קבוע המשמש כתעודת הזהות הקריפטוגרית של בכור בעולמות סיגנל, ובין היתר משמש כדי להוכיח את הזהות שלו.

Signed PreKey - מפתח זמני (מתחדש אחת לכמה זמן) אשר נחתם על ידי מפתח הזהות הפרטי של בכור (Private Identity Key). תפקידו העיקרי של מפתח זה הוא להוכיח את האותנטיות של ה-PreKeyBundle הרלוונטי ולמנוע מצב בו ישתמש ב-PreKeyBundle שלא הועלה לשירות על ידי בכור. (ניתן לוודא כי החתימה אכן בוצעה על ידי ה-Private Identity Key של הישות איתה נרצה לתקשר).

One-Time PreKey - מפתח חד פעמי המשמש ליצירת סשן אחד בלבד! (בפועל, בכור יעלה מספר גדול של One-Time PreKeys לשירות (כ-100 במספר) כדי לאפשר למשתמשים שונים לייצר איתו סשן. האפליקציה תוודא אחת לכמה זמן שכמות ה-One-Time PreKeys שבשרת לא נמוכה מדי ולא תמנע יצירה של סשנים עתידיים, אך גם במקרה זה יש פתרון, עליו לא נדבר במאמר זה).

לאחר שאבי יבקש את ה-**One-Time PreKey** הרלוונטי, הוא ימחק מהשרת.

X3DH In Practice

לאחר שאבי קיבל את ה-PreKeyBundle של בכור ווידא את האותנטיות שלו (בדק את חתימת ה- Signed PreKey), עליו לייצר זוג מפתחות זמני (Ephemeral KeyPair) וכעת הוא מוכן לבצע X3DH. בפועל, X3DH הוא שרשור של מספר חישובי DH (DH1 מייצג את ערך ה-DH הראשון שנחשב, DH2 את השני וכן האלה):

$$DH1 = DH(IK_A, SPK_B)$$

מורכב ממפתח הזהות הפרטי של אבי (יוצר הסשן) וה-Signed PreKey של בכור (הצד המקבל).

$$DH2 = DH(EK_A, IK_B)$$

מורכב מהמפתח הפרטי הזמני של אבי (אשר נוצר עבור סשן זה בלבד) ומה-Public Identity Key של בכור.

$$DH3 = DH(EK_A, SPK_B)$$

מורכב מהמפתח הפרטי הזמני של אבי, ומה-Signed PreKey של בכור.

$$DH4 = DH(EK_A, OPK_B)$$

מורכב מהמפתח הפרטי הזמני של אבי, ומה-One-Time-PreKey של בכור.

- במצבים מסוימים, בהם נגמרה אספקת ה-One-Time PreKeys על השרת (לדוגמא במצב בו הצד המקבל לא התחבר לאפליקציה תקופה ולא העלה מפתחות חדשים לשרת) החישוב של DH4 לא יבוצע, וה-X3DH יבוצע על סמך 3 ערכים, ולא 4.

נשרשר את ערכי ה-DH שקיבלנו, ולבסוף נעביר אותם דרך פונקציית KDF:

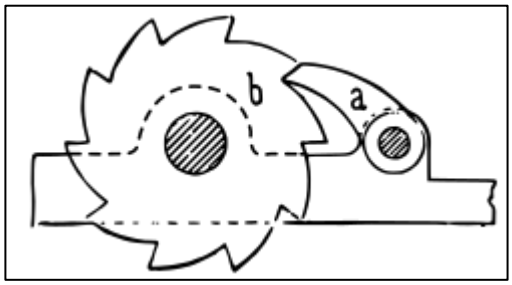
$$SK4 = KDF(DH1 \,||\, DH2 \,||\, DH3 \,||\, DH)$$

התוצאה שנקבל מכונה SK (Session Key) והיא משמשת כקלט הראשוני עבור אלגוריתם Double Ratchet. כפי שציינו, X3DH הוא הדרך של שני משתמשים לייצר ביניהם סשן ראשוני (ומכאן השם של הפלט שלו, Session Key) והשימוש בו נעשה בפעם הראשונה שהם מתחילים לתקשר. לאחר שסיימו לבצע X3DH בהצלחה, האחריות עוברת ל-Double Ratchet.

Double Ratchet

האלגוריתם מתבסס על שני מנגנונים דמויי גלגל שיניים במהותם (או ליתר דיוק מחגר משנן- מנגנון המאפשר תנועה סיבובית או קווית של חלק מכונה בכיוון אחד בלבד ומונע את תנועתו בכיוון ההפוך), המשמשים לייצור והחלפה של מפתחות.

אך לפני שנבין את מנגנונים אלה, נבין כיצד האלגוריתם עובד ב-High Level.



כאשר שני משתמשים מתקשרים מעל סיגנל ומבצעים X3DH כדי להתחיל סשן, הם ישתמשו ב-Session Key שיצרו, כדי בסופו של דבר לגזור מפתחות סימטריים לטובת הצפנה ופענוח של הודעות.

אך כפי שכבר הבנו, השימוש במפתחות אסימטריים קבועים לא בטוח ולכן בסיגנל המשתמשים יחדשו את המפתחות האסימטריים שלהם אחת לכמה הודעות.

בפועל, חידוש המפתחות מתבצע בכל פעם שצד מסוים שולח הודעה ראשונה אחרי שקיבל כבר הודעה מהצד השני:

הצד שמחדש את המפתחות שלו יצרף את המפתח הפומבי החדש להודעה ששלח, כך שהצד השני ישתמש בו כדי לבצע חישוב DH, שבסופו של דבר יאפשר לו לפענח את תוכן ההודעה. כאשר המשתמש שזה עתה קיבל את ההודעה ישיב למשתמש ששלח לו אותה, גם הוא יחדש את המפתחות באותה צורה שזה עתה תיארנו.

KDF chains

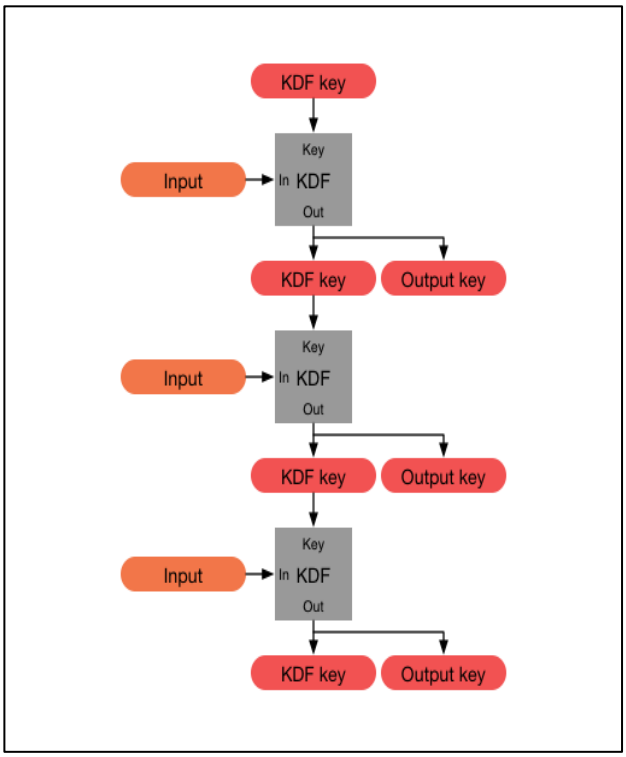
כדי להבין את האלגוריתם בצורה מיטבית, עלינו תחילה להבין קונספט ליבתי עליו מתבסס האלגוריתם המכונה KDF Chain, או בעברית שרשרת KDF.

בגדול, שרשרת KDF היא מנגנון קריפטוגרפי שבו פונקציית KDF בעלת מספר פלטים משורשת לעצמה בצורה הבאה:

הפלט הראשון הוא KDF key / Chain key, מפתח סודי המשתנה בכל איטרציה של השרשרת, חיוני ליצירת השלב הבא בשרשרת (למעשה ישמש כקלט בשימוש הבא בפונקציית ה-KDF).

הפלט השני הוא Output key, שהוא המפתח המשמש להצפנה כללית.

שרשרת שכזו מאפשרת לנו לייצר מפתחות הצפנה (Output keys) אותם ניתן לייצר אך ורק בהינתן ה- KDF key, אשר נשאר סודי לאורך התהליך.



התכונות המרכזיות של שרשרת KDF הן:

- גמישות - למי שאין ידע על מפתח ה-KDF, לא יוכל להבדיל בין הפלטים השונים של פונקציית ה-KDF, ולמעשה לא יכול להבדיל בין ה-KDF Key, ל-Output key.
- סודיות קדימה - גורם שהצליח להשיג את ה-KDF key הנוכחי, לא יוכל לחשב מפתחות קודמים "התאוששות" **(Break-in recovery)** - המנגנון יודע לרפא את עצמו במידה וה-KDF Key נחשף.

על הנייר, נראה שאם גורם שלישי משיג את ה-KDF Key, שום דבר לא ימנע ממנו לחשב את המשך השרשרת, אך אלגוריתם Double Ratchet פותר בעיה זו בעזרת מנגנון עליו נלמד כעת.

(בעבר, פרוטוקול סיגנל היה מכונה "**Axolotol**" על שם סלמנדרה מקסיקנית שיכולה "לחדש" חלקים מגופה - https://signal.org/blog/signal-inside-and-out/)

Diffie-Hellman ratchet

מטרת מנגנון זה היא חידוש והחלפה מתמשכים של מפתחות אסימטריים עליהם מתבססים המפתחות הסימטריים המשמשים להצפנה ופענוח של הודעות.

הכל מתחיל אחרי X3DH, שבפועל עושה קצת יותר מלייצר Session Key.

בפועל, הפלט של X3DH משמש כ-Seed עבור Double Ratchet, ובאופן ספציפי כקלט הראשון בשרשרת ה-KDF (ה-KDF key הראשון) עליה המנגנון Diffie-Hellman ratchet מתבסס.

בקונטקסט של מנגנון זה, ה-**KDF key** בשרשרת ה-**KDF** מכונה **Root Key**.

כאשר מתרחשת החלפת מפתחות מכיוון אחד המשתמשים, מתבצעת יצירה של חוליה חדשה בשרשרת וזאת על ידי שימוש בפונקצית KDF המקבלת 2 ערכים ומחזירה 2 ערכים:

(RK_new, CK_sending) = **KDF_RK**(RK_old, DH_output)

- **RK_old** - ה-Root Key הנוכחי (Session Key הוא למעשה Root Key לכל דבר ועניין, ואף אפשר להתייחס אליו כ-RK_0, שמו המיוחד נובע מהיותו הראשון בסשן).
- **DH_output** - תוצאת ה-DH שהתקבלה על בסיס המפתחות החדשים.
- **RK_new** - ה-Root Key החדש.
- **CK_sending** - ה-**Chain Key** הנוכחי, שאת משמעותו נבין מיד.

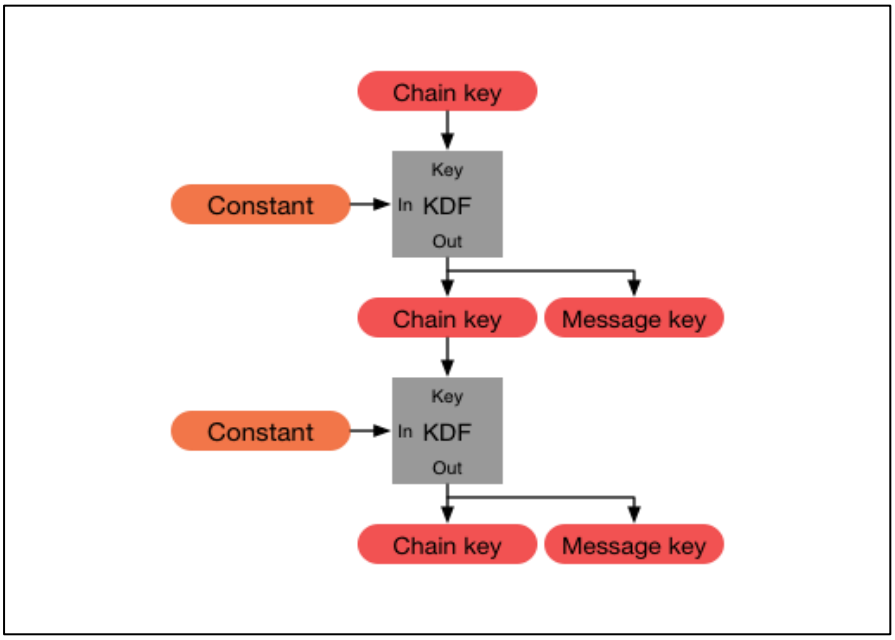
כאשר נקבל את ערך ה-**Root Key** החדש, נמחק באופן מידי את ערך ה-**Root Key** הישן, וזאת על מנת למנוע מגורם שלישי לחדש חוליות ישנות יותר בשרשרת.

Symmetric-key ratchet

המנגנון השני באלגוריתם, המכונה Symmetric-key ratchet, הוא המנגנון האחראי לייצר מפתחות סימטריים לצורך הצפנה ופענוח של הודעות.

בשונה ממנגנון ההצפנה עליו דיברנו במאמר הקודם, בו משתמש מגריל מפתח AES כדי להצפין הודעה, מפתחות ההצפנה הסימטריים בסיגנל המשמשים להצפנה ופענוח של הודעות מיוצרים בעזרת שרשרת KDF, בה ה-**KDF Key** הוא למעשה ה-**Chain Key** שמיוצר על ידי ה-**Diffie Hellman ratchet**. שרשרת ה-KDF עליה מתבסס המנגנון פשוטה מאד:

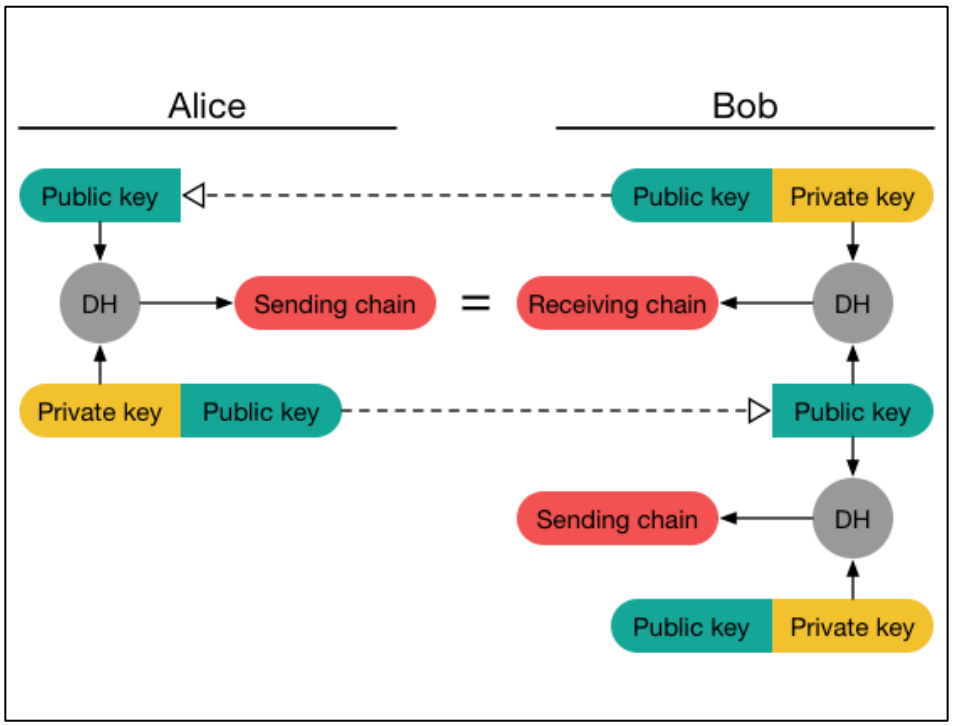
הקלט הראשוני הוא ה-Chain Key שמיוצר באיטרציה של Diffie Hellman ratchet, ובכל איטרציה של השרשרת, נוצר **Chain Key** חדש, ו-**Message Key** המשמש להצפנה / פענוח של הודעה בודדת:



.

מאחר ושני הצדדים בסשן משתמשים באותו Chain Key התחלתי כדי לייצר את שרשרת ה-KDF, הם גם מייצרים Message Keys זהים בכל איטרציה, דבר המאפשר להם לחלוק מפתחות לצורך הצפנה ופענוח, מבלי לשלוח את המפתחות הללו אחד לשני. בפועל, כאשר צד A מחדש את המפתחות שלו ומקדם את מנגנון ה-Diffie Hellman ratchet בשלב, הוא מייצר שרשרת KDF המכונה Receiving chain המשמשת אותו לפענוח של הודעות. לעומת זאת, כאשר צד B יגלה שבוצעה התקדמות בשרשרת, הוא יצור שרשרת KDF זהה לזו שיצר משתמש A, המכונה Sending chain, רק שזו תשמש אותו להצפנה של הודעות.

(בפועל, מאחר והמפתחות המיוצרים על ידי השרשראות זהים, המשתמשים יכולים להשתמש ב- Sending & Receiving chains גם להצפנה וגם לפענוח של הודעות!)



שרשראות אלה ישמשו את הצדדים כדי להצפין ולפענח הודעות, עד אשר אחד הצדדים יחדש את המפתחות שלו, מה שיגרום להתקדמות שרשרת ה-Diffie Hellman וכתוצאה מכך ליצירה של Sending & Reciving chain חדשים.

בשונה ממנגנון ההצפנה שהכרנו במאמר הקודם, בו משתמש מגריל מפתח AES אקראי ומשתמש בו כדי להצפין הודעה, בסיגנל המשתמשים מייצרים מפתחות על סמך סוד משותף מתחדש ומעדכנים אחד את השני איזה מפתח הצפין איזה הודעה על סמך האינדקס של המפתח בשרשרת ה-KDF הנוכחית.

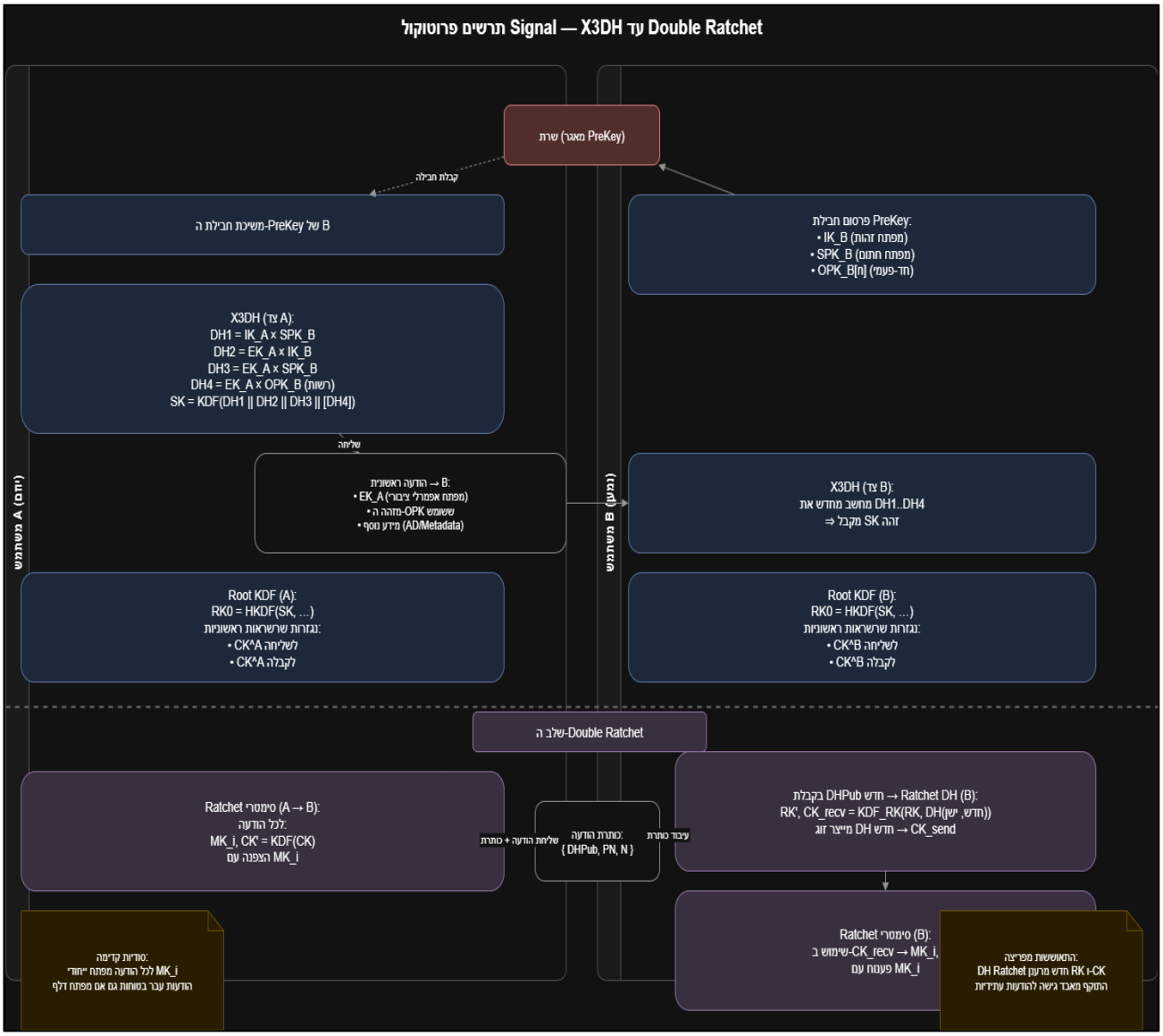
מנגנון זה מוודא שמפתח בודד משמש להצפנה ופענוח של הודעה בודדות, ואף מוודא שלא ניתן להשיג גישה למפתחות ישנים יותר (גם כי הם נמחקים, וגם כי הם מחושבים על בסיס פונקציית KDF אותה לא ניתן להפוך). החיסרון היחיד במנגנון זה הוא שבהינתן מפתח X ו-Root Key, נוכל לחשב את המפתח X+1 בשרשרת. אך בעיה זו נפתרת על ידי Diffie-Hellman ratchet, אשר מוודא שה-Root Key הנוכחי מתחדש לעיתים תכופות, כך שגם אם הגיע לידיים הלא הנכונות, לאחר מספר הודעות הוא כבר לא יהיה רלוונטי.

לאחר שהכרנו את המנגנונים העיקריים המרכיבים את הפרוטוקול, בואו נחזור עליהם:

X3DH- פרוטוקול החלפת מפתחות המאפשר לשני משתמשים לייצר סשן אחד עם השני על סמך מפתחות הצפנה המאוחסנים בשרתים של סיגנל, דבר המאפשר לשני המשתמשים להתחיל סשן גם אם אחד מהם אינו מחובר. הפרוטוקול משתמש במספר מפתחות הצפנה שונים ואף מבצע בדיקת זהות (Identity Keys, Signed PreKey) כדי לייצר מפתח הנקרא Session Key אשר משמש כקלט הראשוני של Double Ratchet.

Double Ratchet - אלגוריתם המאפשר לשני משתמשים לייצר מפתחות הצפנה סימטריים להצפנה ופענוח של הודעות, תוך חידוש מתמשך של המפתחות האסימטריים עליהם הם מתבססים. האלגוריתם מורכב מ-2 שרשראות KDF, האחת (Diffie-Helman ratchet) אחראית על חידוש מפתחות אסימטריים, והשנייה (Symmetric key ratchet) אחראית על יצירת מפתחות סימטריים על בסיס מפתח סודי המיוצר על ידי השרשרת הראשונה.

ניתן להסתכל על השילוב בין השתיים כלולאת for מקוננת, כך שעל כל איטרציה של Diffie-Helman ratchet, תתבצע אחת או יותר איטרציות של Symmetric key ratchet.



כעת, לאחר שהכרנו את הפרוטוקול קצת יותר, נחזור לבעיות שהצפנו בתחילת המאמר ונבין איך סיגנל פותרת אותם:

- מפתחות אסימטריים סטטיים - בעזרת Double Ratchet, סיגנל מאפשרת ל-2 משתמשים לחדש מפתחות אסימטריים לעיתים תכופות, גם אם אחד המשתמשים אינו מחובר לשירות באותם רגעים.
- שימוש במפתחות אסימטריים עבור יותר מסשן אחד - סיגנל פותרת בעיה זו בכך שהיא מאפשרת למשתמש להעלות מפתחות הצפנה חד פעמיים לשירות, כאלה המאפשרים למשתמשים לייצר ביניהם סשן בהתבסס על פרמטרים קיפטוגרפיים יחודיים עבור אותו סשן, לדוגמא, על ידי One-Time PreKey - מפתח הצפנה בו נעשה שימוש ליצירת סשן אחד בלבד.

מקורות מידע וקריאה נוספת

- תיעוד מעמיק של פרוטוקול סיגנל - https://signal.org/docs/
- הספרייה הרשמית המממשת את הפרוטוקול - https://github.com/signalapp/libsignal

סיכום

במאמר זה אתגרנו את מנגנון ההצפנה שעליו דנו במאמר הקודם במטרה להבין את המורכבות הניכרת ביצירת פרוטוקול הצפנה אשר עומד בסטנדרטים מודרניים.

הכרנו את ההיסטוריה של אפליקצית סיגנל ואת ליבת פרוטוקול ההצפנה שלה (X3DH, Double Ratchet), אשר פותר בצורה אלגנטית את הבעיות שהוצגו בתחילת המאמר, עובדה אשר עודדה מספר רב של אפליקציות לאמץ את הפרוטוקול לחיקיהן.