התקפות על מערכת ההצפנה RSA

Attacks on the RSA Cryptosystem

עודד בראש

מסמך זה הורד מהאתר http://www.underwar.co.il

תוכן העניינים

1. מבוא

מושג ההצפנה באמצעות מפתח פומבי נוצר בשנת 1977 על ידי Diffie & Hellman [1][1]. בשנה שלאחר מכן, Rivest, Shamir & Adleman [2], בנו מימוש של הרעיון בדמות מערכת ההצפנה RSA. כיום ניתן למצוא יישומים של RSA בטלפונים, בכרטיסים חכמים (smart cards) ובכרטיסי גישה לרשתות מקומיות. כמוכן, RSA נכללת במרבית הפרוטוקולים לתקשורת בטוחה באינטרנט. מערכת ההצפנה RSA מופיעה, למעשה, בכל מקום בו נדרשת הבטחה של מידע דיגיטלי.

מאז פרסומה הראשון, מערכת ההצפנה RSA נבחנה על ידי חוקרים רבים ונמצאו מספר התקפות מרתקות, אולם אף אחת מהן אינה הרסנית וניתן להימנע מהן על ידי מימוש נכון של המערכת. בעבודה זאת אני אתן סקירה כללית על דרך הפעולה של RSA. אני אדון בגישות ובאלגוריתמים השונים אשר פותחו לצורך פיצוח מערכת הצפנה זאת, וכן אתאר את מידת הצלחתם בכך וכיצד ניתן להתגונן מפניהם. אני אבדוק עד כמה המערכת בטוחה כיום, ואנסה להעריך עד מתי המערכות בהן משתמשים היום יישארו בטוחות. בנוסף, אני אתאר מספר התפתחויות טכנולוגיות עדכניות אשר עשויות לסכן את המערכת.

התקפות על מערכת ההצפנה RSA ניתנות לחלוקה לשלושה סוגים. הסוג הראשון מתאר "התקפות אלמנטריות", אשר מסתמכות על שימוש או מימוש לא בטוח של מערכת ההצפנה (למשל, על ידי ניצול של מקרים פרטיים). שנית, קיימות התקפות מתוחכמות יותר, המסתמכות על אלגוריתמים והיוריסטיקות כדי לפתור את הבעיה הקשה מבחינה חישובית עליה מערכת ההצפנה מבוססת. במקרה של RSA, התקפה כזאת יכולה להיות אלגוריתם פירוק לגורמים מהיר. לבסוף, קיימות התקפות המתבססות על שימוש בחומרה ייעודית לביצוע פירוק לגורמים (כדוגמת התקן ה-TWINKLE או מחשוב קוונטי).

[1] מספרים הנמצאים בין סוגריים מרובעים מציינים מספר מקור [מאמר או ספר] ברשימת המקורות.

2. מערכת ההצפנה RSA

הצפנה באמצעות מפתח פומבי לעומת הצפנה באמצעות מפתח סודי

בהצפנה עם מפתח סודי[2] (secret-key cryptography), השולח והמקבל של ההודעה מכירים ומשתמשים באותו מפתח סודי; השולח משתמש במפתח הסודי להצפנת ההודעה והמקבל משתמש באותו מפתח סודי על מנת לפענח את ההודעה. הקושי העיקרי הוא הסכמה על מפתח סודי מתאים בין שולח ומקבל ההודעה מבלי שאף אחד יגלה אותו. אם השולח והמקבל רחוקים אחד מהשני, עליהם לסמוך על שליח, קו טלפון, או אמצעי אחר להעברת המפתח הסודי המוסכם. מכיוון שעל כל המפתחות במערכת מפתח סודי להישאר חסויים, למערכות אלו יש בעיה קשה בביצוע ניהול מפתחות (key management) בטוח, במיוחד במערכות פתוחות עם מספר רב של משתמשים.

במערכת הצפנה המבוססת על מפתח ציבורי[3] (public-key cryptoSystem) [3], כל משתתף יוצר בעצמו מפתח סודי (secret key) ומפתח ציבורי (public key). כל מפתח הוא פריט מידע המשמש לפעולת ההצפנה או הפענוח. נניח שקיימים שני משתתפים המשתמשים במערכת ההצפנה: "איה" ו"בועז", ונסמן את המפתחות הציבוריים והסודיים שלהם בהתאמה על ידי P_A ו-S_A עבור איה, ו-P_B ו-S_B עבור בועז. כמוכן, נניח שקיים מצותת אשר מנסה לפענח את ההודעות המוצפנות ונכנה אותו בשם "מארווין". כל אחד שומר בסוד את המפתח הסודי שלו, אבל יכול להעביר את המפתח הציבורי לרשות הכלל (למשל, על ידי פרסומו במדריך ציבורי).

נסמן ב-D את קבוצת ההודעות המותרות. לדוגמא, D עשויה להיות קבוצת כל הסדרות באורך סופי של ספרות. כלומר, על כל הודעה טקסטואלית, למשל, הופעלה כבר שיטה ישירה ופשוטה כלשהי להפיכת תווים למספרים. המפתחות, הציבורי והסודי, מגדירים פונקציות שניתן להפעילן על כל הודעה. אנו דורשים שהמפתח הציבורי והמפתח הסודי יגדירו פונקציות חד-חד-ערכיות מ-D לעצמה, כלומר לכל מפתח הצפנה ישנו רק מפתח פענוח יחיד ולהיפך. הפונקציה המתאימה למפתח הציבורי P_A של איה מסומנת על ידי $P_A()$, והפונקציה המתאימה למפתח הסודי שלה S_A מסומנת על ידי $S_A()$. אנו מניחים שהפונקציות $P_A()$ ו-$S_A()$ קלות לחישוב בהינתן המפתחות המתאימים P_A ו-S_A.

שני המפתחות של כל משתתף מגדירים פונקציות הופכיות זו לזו, כך שניתן להצפין עם אחד ולפענח עם האחר ולהיפך. עבור כל הודעה $M \in D$ מתקיים:

$$M = S_A(P_A(M))$$

$$M = P_A(S_A(M))$$

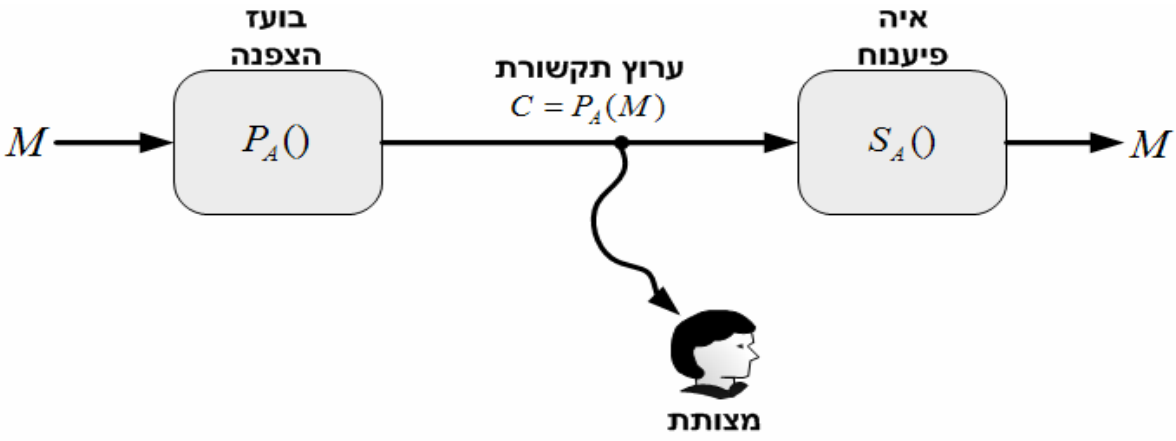
[2] נקראת גם הצפנה סימטרית (symmetric cryptography).
[3] נקראת גם הצפנה אסימטרית (asymmetric cryptography).

כלומר, אם מפעילים על M את שני המפתחות P_A ו-S_A בזה אחר זה, בסדר כלשהו, מקבלים בחזרה את M.

כדי לשלוח הודעה לאיה, משתמש בועז בפונקציית ההצפנה הציבורית של איה, $P_A()$. כעת יכולה איה לפענח את ההודעה בעזרת הפונקציה הפרטית שלה, $S_A()$. מערכת הצפנה הפועלת באמצעות מפתח ציבורי מסתמכת על ההנחה שרק איה מסוגלת לחשב את $S_A()$ בזמן סביר. אם איה תחשוף את המפתח הסודי S_A, כל אחד יוכל לפענח את ההודעות המוצפנות שלה ומערכת ההצפנה לא תוכל לספק את ההגנה הדרושה. תנאי בסיסי הוא שהפונקציות תהיינה כאלה שלא תאפשרנה להסיק שיטה לחישוב $S_A()$ מתוך פונקציית ההצפנה $P_A()$. כלומר, אף על פי שכולם מסוגלים לחשב ביעילות את $P_A()$, רק איה מסוגלת לחשב בצורה יעילה את הפונקציה ההפכית $S_A()$. במילים אחרות, בלא ידיעת המפתח הסודי S_A, החישוב של $S_A()$ הוא "לא מעשי מבחינה חישובית". הקושי העיקרי בתכנון מערכת הצפנה המבוססת על מפתח ציבורי הפועלת היטב הוא למצוא כיצד ליצור מערכת שבה ניתן לחשוף טרנספורמציה $P_A()$ בלי לחשוף בכך כיצד לחשב את הטרנספורמציה ההפכית המתאימה $S_A()$.

נתאר כעת את סדר הפעולות הדרוש לביצוע הצפנה במערכת המבוססת על מפתח ציבורי (ראה איור 1). נניח שבועז רוצה לשלוח לאיה הודעה M, מוצפנת כך שתיראה לכל מצותת כקשקוש בלתי קריא. משלוח ההודעה מתנהל באופן הבא:

- בועז מקבל את המפתח הציבורי P_A של איה (ממדריך ציבורי או ישירות מאיה).
- בועז מחשב את כתב-הצופן (ciphertext) $C = P_A(M)$ של ההודעה M ושולח את C לאיה.
- כאשר איה מקבלת את כתב הצופן C, היא מפעילה עליו את המפתח הסודי שלה S_A ומקבלת את ההודעה המקורית: $M = S_A(C)$.



איור 1

איה יכולה לחשב את M מתוך C מכיוון ש-$S_A()$ ו-$P_A()$ הן פונקציות הפכיות. רק איה יכולה לחשב את M מתוך C, שכן רק היא יכולה לחשב את $S_A()$. הצפנת M באמצעות $P_A()$ מגנה על M מפני גילוי על ידי כל אדם אחר פרט לאיה.

מערכות הצפנה באמצעות מפתח פומבי פותרות את הבעיה של מערכות מפתח סודי בכך שאין צורך להפיץ מפתח סודי בין הצדדים המתקשרים.

<u>מערכת ההצפנה RSA</u>

מערכת ההצפנה באמצעות מפתח ציבורי RSA מבוססת על ההבדל הדרמטי בין הקלות אשר בה ניתן למצוא מספרים ראשוניים גדולים לבין הקושי לפרק לגורמים את המכפלה של שני ראשוניים גדולים.

במערכת ההצפנה RSA (RSA cryptosystem), כל משתתף יוצר את המפתח הציבורי והמפתח הסודי באמצעות האלגוריתם הבא:

1) בחר באקראי שני מספרים ראשוניים גדולים, p ו-q. הראשוניים p ו-q יהיו, נאמר, בני 100 ספרות עשרוניות כל אחד.
2) חשב את N באמצעות המשוואה $N = pq$.
3) בחר שלם אי-זוגי קטן e שהוא זר ל-$\phi(N)$, כאשר $\phi(N)$ היא פונקצית אוילר. מהגדרת N ברור כי $\phi(N) = (p-1)(q-1)$.
4) חשב את d, ההפכי הכפלי של e, מודולו $\phi(N)$[4].
5) פרסם את הזוג $P = (e, N)$ כמפתח RSA הציבורי שלך (RSA public-key).
6) שמור בסוד את הזוג $S = (d, N)$ כמפתח RSA הסודי שלך (RSA secret-key).

עבור מערכת זו, התחום D הוא הקבוצה Z_N. הטרנספורמציה של הודעה M המתקבלת מהפעלת מפתח ציבורי $P = (e, N)$ היא:

$$P(M) = M^e \pmod{N}$$

הטרנספורמציה של כתב צופן C באמצעות מפתח סודי $S = (d, N)$ היא:

$$S(C) = C^d \pmod{N}$$

משוואות אלה מתאימות להצפנה וגם לחתימה. ליצירת חתימה, החותם מפעיל את המפתח הסודי שלו על ההודעה שעליה הוא רוצה לחתום, ולא על כתב-צופן. כדי לאמת חתימה, מפעילים על

[4] קיים d כזה והוא יחיד שכן e זר ל-$\phi(N)$.

החתימה, ולא על הודעה המיועדת להצפנה, את המפתח הציבורי של החותם (ראה סעיף "שימושים במערכת ההצפנה RSA").

ביצוע הצפנה, פענוח או חתימה באמצעות RSA כרוכה בביצוע פעולת העלאה בחזקה מודולרית. פעולה זו מתבצעת באמצעות סדרה של הכפלות מודולריות. סיבוכיות הזמן בהפעלת מפתח סודי היא $O(k^3)$, כאשר k הוא מספר הסיביות במודולוס.

הכפלת גודל המודולוס N פי שניים מאריכה את הזמן הדרוש להצפנת הודעה ובדיקה של חתימה, בממוצע, פי ארבעה, ואת הזמן הדרוש לפענוח הודעה וביצוע חתימה פי שמונה. הזמן הדרוש ליצירת המפתח גדל פי 16.

על אף שהיעילות של מערכות RSA הולכת ומשתפרת בהתמדה, עדיין מערכות RSA איטיות באופן משמעותי ממערכות הצפנה סימטריות סטנדרטיות כמו DES. באופן כללי DES מהירה פי 100 לפחות מ-RSA[5].

<u>משפט 1.1 (נכונות RSA)</u>

המשוואות $P(M) = M^e \pmod N$ ו- $S(C) = C^d \pmod N$ במערכת RSA מגדירות טרנספורמציות הפכיות זו לזו ב- Z_N המקיימות $M = S_A(P_A(M))$ ו- $M = P_A(S_A(M))$. ♦

הוכחת משפט 1.1 (נכונות RSA) מתבססת על משפט אוילר לפיו: לכל a ו- n טבעיים שזרים האחד לשני מתקיים $a^{\phi(n)} = 1 (\mathrm{mod}\, n)$. ההצפנה והפיענוח נעשים בעזרת שני ההפכיים הכפליים המודולריים, e ו- d, מודולו $\phi(n)$. את ההפכיים המודולריים הללו ניתן למצוא בעזרת הגרסה המורחבת של האלגוריתם של אוקלידס. ממשפט אוילר מקבלים, $S_A(C) = C^d = (M^e)^d = M^{ed} = M^{\phi(n)+1} = M (\mathrm{mod}\, n)$.

ברור שבטיחותה של מערכת ההצפנה RSA מתבססת על הקושי לפרק לגורמים מספרים שלמים גדולים, שהרי אם יריב יכול לפרק לגורמים את המודולוס N במפתח הציבורי, אזי מידיעת הגורמים p ו- q הוא יכול לחשב את המפתח הסודי באותו אופן שעשה זאת יוצר המפתח הציבורי.

[5] במימושי חומרה, DES מהירה מ-RSA פי 10^3 עד 10^4 (תלוי באופן המימוש).

<u>תכונות מתמטיות של RSA</u>

לאלגוריתם ההצפנה של RSA שתי תכונות מתמטיות עיקריות:

תכונה 1.2 (תכונת ההדדיות או ההפכיות): $P \circ S = S \circ P = Id$ (the identity function).

תכונה 1.3 (תכונת הכפליות): $\forall x, y \quad S(xy) = S(x)S(y)$.

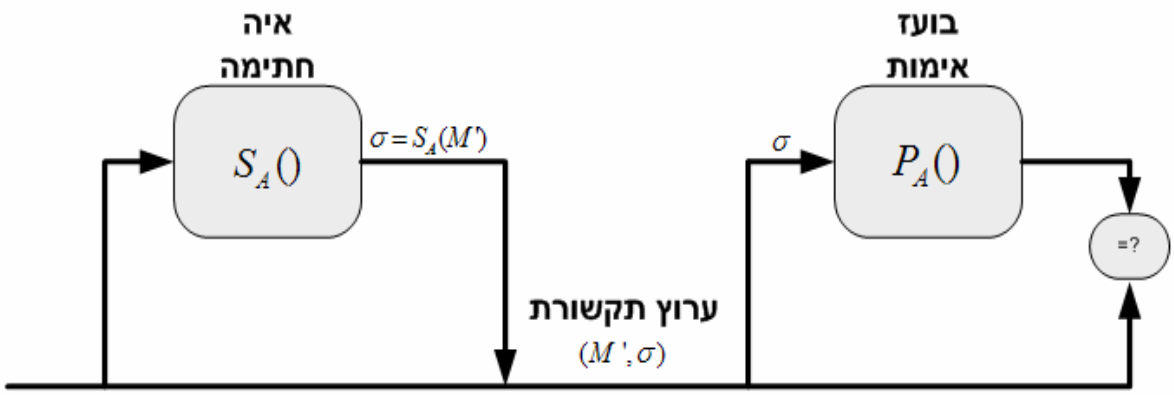
תכונת ההפכיות והכפליות מובילות למספר נקודות חולשה אפשריות כפי שנראה בהמשך.

<u>שימושים במערכת ההצפנה RSA</u>

מערכת RSA היא מערכת ההצפנה של מפתח פומבי הנפוצה ביותר כיום. היא משמשת במגוון מוצרים, פלטפורמות ותעשיות מסביב לעולם. היא משולבת במספר מערכות הפעלה ונכללת בכל הפרוטוקולים העיקריים לתקשורת בטוחה באינטרנט. בחומרה, ניתן למצוא את RSA בטלפונים בטוחים, בכרטיסי רשתות Ethernet, ובכרטיסים חכמים (smart cards). שימוש חשוב של מערכות הצפנה באמצעות מפתח פומבי בכלל ו-RSA בפרט הוא חתימות דיגיטליות. חתימה דיגיטלית היא דרך לאימות היוצר של מסמך אלקטרוני, כתחליף לחתימה בכתב יד.

נניח שאיה רוצה לשלוח לבועז תשובה M' חתומה באופן דיגיטלי (איור 2). החתימה הדיגיטלית מתבצעת על פי התרחיש שלהלן:

- איה מחשבת את **החתימה הדיגיטלית** (digital signature) שלה σ עבור ההודעה M' באמצעות המפתח הסודי שלה S_A והמשוואה $\sigma = S_A(M')$.
- איה שולחת לבועז את הזוג (M', σ) של ההודעה והחתימה.
- כאשר בועז מקבל את (M', σ), הוא יכול לאמת שההודעה נשלחה על ידי איה באמצעות אימות המשוואה $M' = P_A(\sigma)$ תוך שימוש במפתח הציבורי של איה (אנו מניחים ש-M' מכילה את שמה של איה כך שבועז יודע באיזה מפתח ציבורי להשתמש). אם המשוואה אינה מתקיימת, בועז מסיק שההודעה M' או החתימה נפגמו בשל שגיאות תמסורת או שהזוג (M', σ) הוא ניסיון זיוף.



איור 2

חתימה דיגיטלית מאמתת הן את זהותו של החותם והן את תוכן ההודעה החתומה ולכן היא מקבילה לחתימה בכתב יד בסופו של מסמך כתוב. החתימה הדיגיטלית ניתנת לאימות על ידי כל מי שיש לו גישה למפתח הציבורי של החותם. נשים לב כי הודעה חתומה אינה מוצפנת; ההודעה גלויה לעין ואינה מוגנת מחשיפה. על מנת ליצור הודעות גם חתומות וגם מוצפנות ניתן לבצע הרכבה של פרוטוקולי ההצפנה והחתימה שתוארו לעיל. החותם מוסיף תחילה את חתימתו הדיגיטלית בסוף ההודעה, ואז מקודד את הזוג הודעה/חתימה המתקבל באמצעות המפתח הסודי שלו, ומקבל את ההודעה המקורית וגם את החתימה הדיגיטלית המצורפת אליה. לאחר מכן, הוא יכול לאמת את החתימה באמצעות המפתח הציבורי של החותם.

חלק מההתקפות שאתאר בעבודה זאת מכוונות במיוחד לחתימות דיגיטליות באמצעות RSA.

<u>בעיות "קשות"</u>

מרכיב מרכזי במערכות הצפנה אסימטריות הוא קיומה של בעיה "קשה", כלומר בעיה אשר לא מעשי מבחינה חישובית לפתור אותה. בפרט, בעיות רלוונטיות במיוחד הן בעיות שלמות ב-NP (NP-complete)[6] אשר משערים שלא ניתן למצוא אלגוריתם פולינומיאלי אשר פותר אותן. דוגמאות לבעיות קשות: פירוק לגורמים, בעיית הלוגריתם הדיסקרטי, בעיית הסוכן הנוסע, בעיית סיפוק של ביטויים בוליאניים (SAT problem), בעיית צביעת גרף, בעיית תרמיל הגב (Knapsack) ועוד.

השימוש בבעיה "קשה" במערכות הצפנה אסימטריות נעשה באמצעות פונקציות חד-כיווניות (one-way function) ופונקציות מלכודת חד-כיווניות (one-way trapdoor function). פונקציה חד-כיוונית היא פונקציה אשר ניתן לחשב אותה בקלות בכיוון אחד ובלתי מעשי מבחינה מעשית לחשב אותה בכיוון השני. פונקצית מלכודת חד-כיוונית היא פונקציה חד-כיוונית אשר מאפשרת באמצעות מידע מסוים לחשב בקלות גם את הכיוון ההפוך (אך לא ניתן אם המידע לא קיים).

במערכת RSA משתמשים בפונקצית המלכודת הבאה: $x \mapsto x^e \pmod N$. בהינתן d, הפונקציה ניתנת להיפוך בצורה יעילה.

<u>יצירת מספרים ראשוניים מתאימים</u>

כמו שראינו, RSA דורשת יצירה של מספרים ראשוניים גדולים "אקראיים". מספרים ראשוניים גדולים הם נפוצים למדי, כך שלא נדרש זמן ארוך מידי לבדיקת שלמים אקראיים בגודל המתאים עד שמוצאים ביניהם מספר ראשוני. נסמן ב-$\pi(n)$ את פונקצית התפלגות הראשוניים (prime distribution function) המציינת את מספר הראשוניים הקטנים מ-n או שווים לו.

[6] ואף על פי כן, לא נמצא שימוש נרחב לבעיות אלו במערכות הצפנה של מפתח פומבי.

משפט 1.4 (משפט המספרים הראשוניים)

$$\lim_{n\to\infty}\frac{\pi(n)}{n/\ln(n)}=1$$

♦

ממשפט זה ניתן ללמוד למשל כי מספר הראשוניים באורך 512 ביט או פחות הוא 10^{150}, יותר ממספר האטומים ביקום.

ממשפט המספרים הראשוניים ברור שההסתברות ששלם n אשר נבחר באקראי יהיה ראשוני היא $1/\ln(n)$. לכן, כדי למצוא ראשוני השווה לאורכו של n, יש לבחון בקירוב $\ln(n)$ שלמים הקרובים ל-n שנבחרו באקראי. למשל, כדי למצוא ראשוני בן 1000 ספרות עלולה להידרש בדיקה של כ-$\ln(10^{100})\approx 2300$ שלמים בני 1000 ספרות שנבחרו באקראי (ניתן לצמצם למחצית את מספר הבדיקות אם בוחרים רק שלמים אי-זוגיים). בנספח א' מוצגת תוכנית פשוטה בשפת C אשר יוצרת מספרים ראשוניים עד למספר נתון.

בדיקת ראשוניות שייכת הן ל-NP והן ל-co-NP. לא ידוע על קיום אלגוריתם לבדיקת ראשוניות שזמן ריצתו פולינומיאלי (כלומר, לא ידוע האם בדיקת ראשוניות שייכת ל-P), וייתכן שבעיה זו אכן מעריכית. בניגוד לבעיות רבות אחרות אשר עבורן לא ידועים פתרונות יעילים מבחינה חישובית, כגון הבעיות השלמות ב-NP, כאן אין לנו סיבה להניח שלא קיימים פתרונות כאלה. למעשה, קיים אלגוריתם לבדיקת ראשוניות שזמן ריצתו פולינומיאלי, אולם נכונותו מסתמכת על השערה מתמטית, עמוקה אך בלתי מוכחת, הנקראת היפותזת רימן המורחבת (extended Riemman hypothesis). אם תוכח אי פעם נכונותה של היפותזה זו, תוכח בכך גם שייכותה של בעיית הראשוניות ל-P. בנוסף, קיימים אלגוריתמים לבדיקת ראשוניות גם בלי להסתמך על היפותזת רימן, שזמן ריצתם "כמעט" פולינומיאלי[7]. באופן מעשי, על ידי שימוש באלגוריתמים הסתברותיים ניתן ליצור אלגוריתמים פולינומיאליים מהירים ביותר לבדיקת ראשוניות וזאת השיטה אשר בה משתמשים בפועל[8].

לא כל מספר ראשוני מתאים לשימוש על ידי מערכת RSA. לכן, נוהגים להבחין בין מספרים ראשוניים **חזקים וחלשים**. מספרים ראשוניים חזקים הם בעלי תכונות אשר גורמות לכך שקשה לפרק לגורמים את המכפלה $N=pq$ על ידי שיטות פירוק לגורמים ידועות (ראה פרק 3: "RSA ופירוק לגורמים"). באופן כללי, הכללים הבאים חייבים להתקיים בעת בחירת המספרים הראשוניים p ו-q:

[7] זמן הריצה של האלגוריתם המהיר ביותר המוכר כיום לבדיקת ראשוניות על מספר n באורך $\lceil\lg(n+1)\rceil$ הוא $(\lg n)^{O(\lg\lg\lg n)}$, זמן שהוא על-פולינומיאלי אך במקצת.

[8] למשל, האלגוריתמים: 1) מילר ורבין 2) סולווי ושטראסן.

1) p חייב להיבחר כך שבפירוק לגורמים של $p-1$ קיים גורם ראשוני גדול r.

2) $p+1$ גם חייב להיות בעל גורם ראשוני גדול s.

3) $r-1$ חייב להיות בעל גורם ראשוני גדול t.

אותם אילוצים חלים גם על q. בנוסף, p ו- q מוכרחים להיות גדולים, וההבדל ביניהם $|p-q|$ חייב אף הוא להיות גדול.

3. מבוא לקריפטואנליזה

קריפטואנליזה היא ענף בקריפטוגרפיה העוסק בפיצוח והתקפות על צפנים. על מנת ליצור מערכת או אלגוריתם הצפנה בטוח יש לבחון את המערכת תוך שימוש בכלים של הקריפטואנליזה, ובכך לגלות ולמנוע נקודות תורפה אפשריות. זאת הסיבה לכך שמערכות ההצפנה הבטוחות (והאמינות) ביותר הן דווקא אלה שהאלגוריתם שלהן אינו חסוי, אלא חשוף לבדיקה קפדנית של הציבור הרחב. למשל, דרך פעולתם של RSA ו-DES ידועה לכל ונבדקה במשך שנים רבות ולכן שיטות הצפנה אלה נחשבות לבטוחות מאוד. לעומת זאת, מערכת ההצפנה Skipjack מסווגת ולכן נחשבת לפחות בטוחה [6-4].

באופן כללי, התקפה (פסיבית) על מערכת הצפנה היא כל מצב אשר בו מתחילים ממידע מסוים על מערכת ההצפנה, ההודעה המקורית (plaintext) וההודעה המוצפנת (ciphertext) ומסיקים מידע נוסף על ההודעה המקורית.

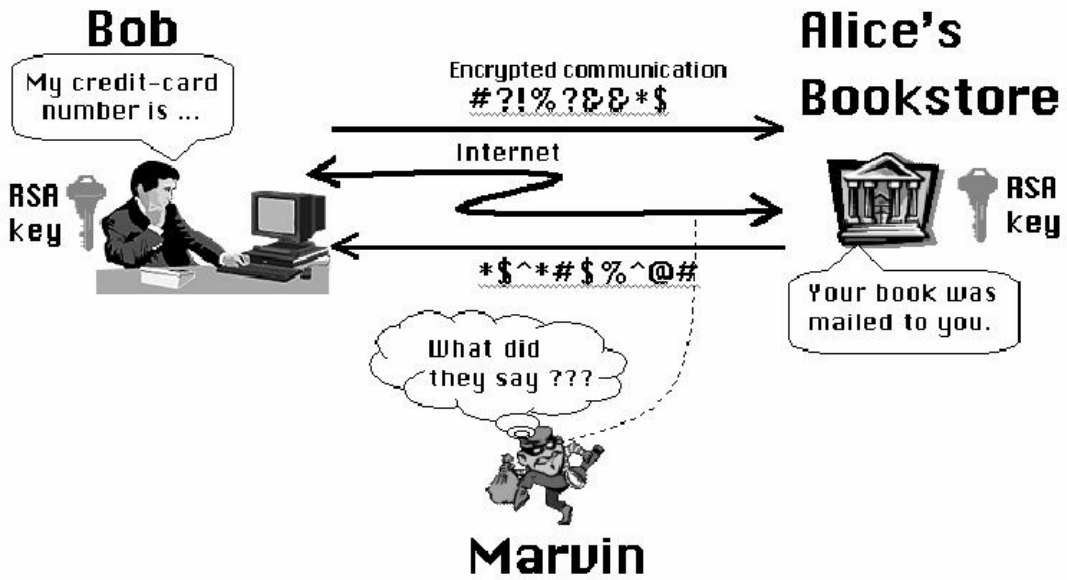
ניתן לחלק התקפות קריפטואנליטיות לארבעה סוגים עיקריים על פי סוג המידע הנמצא ברשות המפענח:

1. **ידיעת ההודעה המוצפנת בלבד** (ciphertext only attack) אשר בה המפענח משיג מדגם של הודעות מוצפנות ללא ההודעות המקוריות המתאימות.
2. **ידיעת ההודעה המקורית** (known plaintext attack) אשר בה המפענח משיג מדגם של הודעות מוצפנות ביחד עם ההודעות המקוריות המתאימות.
3. **הודעה מקורית נבחרת** (chosen plaintext attack) אשר בה המפענח רשאי לבחור מספר הודעות מקוריות וליצור את ההודעות המוצפנות המתאימות.
4. **הודעה מוצפנת נבחרת** (chosen ciphertext attack) אשר בה המפענח רשאי לבחור קטע מהודעה מוצפנת ולנסות למצוא הודעה מתאימה לא מוצפנת.

בעבודה זאת נתרכז בעיקר בהתקפות על פונקצית ה-RSA ולא על מערכת ההצפנה RSA. באופן כללי, הקושי בפריצת אלגוריתם RSA נובע מהקושי להפוך (inverting) את פונקצית ה-RSA עבור קלטים אקראיים. כלומר, בהינתן (N,e,C), תוקף לא יכול לשחזר את ההודעה המקורית M. מערכת הצפנה חייבת להיות חסינה גם בפני התקפות חלשות יותר; בהינתן (N,e,C), אסור שהתוקף יהיה מסוגל לשחזר מידע **כלשהו** על M. חסינות כזאת נקראת **בטיחות סמנטית** (semantic security). לא נדון בנושא הבטיחות הסמנטית בעבודה זאת, רק נציין כי RSA אינה בטוחה סמנטית: בהינתן (N,e,C), ניתן לקבל מידע מסוים על ההודעה M (למשל, אפשר לחשב בקלות את סימן יעקובי [Jacobi symbol] של M מעל N מתוך C). ניתן להפוך את RSA לבטוחה סמנטית על ידי הוספת אקראיות לתהליך ההצפנה.

מידת ההגנה אשר מערכת הצפנה מספקת נקבעת על פי כמות המשאבים (זמן, ידע, מחשוב) הדרושה כדי לפרוץ אותה. כאשר פורץ פונה לפרוץ מערכת מסוימת עליו לבחון את כמות המאמץ הדרושה לו בכדי לממש את הפריצה לעומת התועלת אשר תופק בפועל כתוצאה מהצלחה בפריצה. למשל, אין הגיון בהשקעה של מיליוני דולרים בפענוח מידע השווה מספר אלפי דולרים בלבד. ניתן לספק הגנה טובה יותר על מידע רגיש על ידי שימוש במפתח גדול יותר.

ניתן לתת מספר פירושים למושג "פריצת RSA" (breaking RSA). ההתקפה ההרסנית ביותר היא, כמובן, כזאת אשר בה הפורץ מגלה את המפתח הפרטי המתאים למפתח פומבי נתון; זה מאפשר לפורץ לקרוא כל הודעה שהוצפנה באמצעות המפתח הפומבי, להצפין הודעות ולזייף חתימות. קיימות התקפות אשר מאפשרות פענוח של הודעה יחידה, הצלחה בהתקפה כזאת אינה מאפשרת לפענח הודעות אחרות המשתמשות באותו מפתח. כמובן, קיימות גם התקפות אשר לא מכוונות ל-RSA עצמו אלא למימוש לא בטוח של RSA; אלה לא נחשבים כ"פריצת RSA" שכן הם לא מנצלים חולשה של אלגוריתם RSA, אלא חולשה של מימוש מסוים. לדוגמא, אם משתמש שומר את המפתח הסודי שלו בצורה לא בטוחה, ניתן יהיה לגלות אותו. חשוב להדגיש שכדי שמערכת הצפנה כלשהי ו-RSA בפרט תהיה בטוחה יש להשתמש במימוש בטוח שלה. מדדי בטיחות מתמטיים, כמו בחירת מפתח גדול, לא מספיקים. בפועל, מרבית ההתקפות המוצלחות מכוונות כלפי מימושים לא בטוחים וניהול מפתחות לא מוצלח של RSA.



4. RSA ופירוק לגורמים

כאמור, מערכת ההצפנה RSA מסתמכת על כך שלמרות שניתן לקבוע בקלות יחסית האם מספר N הוא פריק או לא, קשה מאוד לקבוע מהם הגורמים הראשוניים של N.

ראשית, נראה במשפט הבא כי חשיפת המפתח הסודי d ופירוק N לגורמים ראשוניים הן פעולות שקולות. מכאן שאין טעם להסתיר את הפירוק לגורמים של N מכל צד אשר יודע את d.

משפט 4.1

יהי (N,e) מפתח פומבי של RSA. בהינתן המפתח הפרטי d, ניתן לפרק לגורמים באופן יעיל[9] את המודולוס $N=pq$. ולהפך, בהינתן הפירוק לגורמים של N, ניתן לשחזר את d באופן יעיל.

הוכחה: פירוק לגורמים של N נותן את $\phi(N)$. מכיוון ש-e ידוע, ניתן לחשב את d (ההפכי של e, על ידי שימוש באלגוריתם של אוקלידס). בכך הוכחה הטענה ההפוכה.

נראה כעת, כי בהינתן d, ניתן לפרק לגורמים את N. בהינתן d, נחשב את $k=de-1$. מהגדרת d ו-e נובע כי k הוא כפולה של $\phi(N)$. מכיוון ש-$\phi(N)$ זוגי, ניתן לרשום $k=2^t r$ כאשר r אי-זוגי ו-$t\geq 1$. מתקיים $g^k=1$ לכל $g\in z_N^*$, ולכן $g^{k/2}$ הוא שורש ריבועי של היחידה מודולו N. על סמך משפט השאריות הסיני, ליחידה יש ארבעה שורשים ריבועיים מודולו $N=pq$. שניים מתוך השורשים הריבועיים הללו הם ± 1. השניים האחרים הם $\pm x$, כאשר x מקיים $x=1 \bmod p$ וגם $x=-1 \bmod q$. על ידי שימוש באחד מהשורשים הריבועיים האחרונים, ניתן לקבל את הפירוק לגורמים של N על ידי חישוב $\gcd(x-1,N)$. בבירור מתקיים שאם g נבחר בצורה אקראית מ-z_N^*, אזי בהסתברות לפחות חצי, אחד מהאיברים בסדרה $g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \bmod N$ הוא שורש ריבועי של יחידה אשר חושף את הפירוק לגורמים של N. כל איברי הסדרה הנ"ל ניתנים לחישוב באופן יעיל בזמן $O(n^3)$ כאשר $n=\log_2 N$.

♦

כאמור, אם ימצא אלגוריתם פירוק לגורמים יעיל אז RSA אינה בטוחה. הכיוון השני הוא בעיה פתוחה; לא ידוע האם פריצת RSA קשה כמו פירוק לגורמים. כלומר, האם יש לפרק לגורמים את N כדי למצוא את השורש מדרגה e מודולו N. ננסח את הבעיה בצורה מדויקת באופן הבא.

[9] אלגוריתם יעיל הוא אלגוריתם אשר זמן הריצה שלו הוא מסדר גודל $O(n^c)$ כאשר $n=\log_2 N$ ו-c הוא קבוע קטן (פחות מ-5, נניח).

בעיה פתוחה מספר 1

בהינתן שלמים N ו-e המקיימים $\gcd(e,\phi(N))=1$, נגדיר פונקציה $f_{e,N}:Z_N^*\to Z_N^*$ על ידי $f_{e,N}(x)=x^{1/e}$. האם קיים אלגוריתם A, בעל סיבוכיות זמן פולינומיאלית, אשר מחשב את הפירוק לגורמים של N, בהינתן N וגישה לאוראקל $f_{e,N}(x)$ עבור e מסוים?

♦

האוראקל מחשב, עבור כל x, את הפונקציה $f_{e,N}(x)$ ביחידת זמן. לאחרונה [7], נתגלו הוכחות לכך שעבור e קטן פריצת RSA אינה שקולה לפירוק לגורמים והיא למעשה קלה יותר.

בשני העשורים האחרונים, חלו תמורות גדולות בשיטות לביצוע פירוק לגורמים. כיום, אנשים מתעניינים בפירוק לגורמים, בגלל שהוא נחשב כאמת מידה (benchmark) ליכולת חישובית וכשבירת שיאים, ולא רק בגלל חשיבותו בקריפטוגרפיה. שתי פריצות דרך בתחום היו המצאת **הנפה הריבועית** (QS - Quadratic Sieve) בשנות השמונים, **ונפת שדה המספרים** (NFS - Number Field Sieve) באמצע שנות התשעים. טבלה 1 מסכמת את אלגוריתמי פירוק לגורמים העיקריים הקיימים כיום, לצד זמן הריצה המתאים.

אלגוריתם	שנת פרסום	זמן ריצה
Brute force approach	-	$O(e^{Number\ of\ digits\ in\ n})$
Pollard's P-1 method	**1974**	Q
Continued Fraction Method	**1975**	$O(e^{\sqrt{2\ln(n)\ln(\ln(n))}})$
Pollard's Rho Method	**1978**	$\sqrt{P}$
Dixon's Random Squares Method	**1981**	—
Elliptic Curve Method	**1985**	$O(e^{\sqrt{\ln(n)\ln(\ln(n))}})$
Multiple Polynomial Quadratic Sieve Method	**1987**	$O(e^{\sqrt{\ln(n)\ln(\ln(n))}})$
Number Field Sieve	**1988 (further developed 1994)**	$O(e^{(c+o(1))n^{1/3}\log^{2/3}n})$ $c\approx 2$

Q – הגורם הגדול ביותר של $P-1$.

P – הגורם הגדול ביותר של N.

טבלה 1

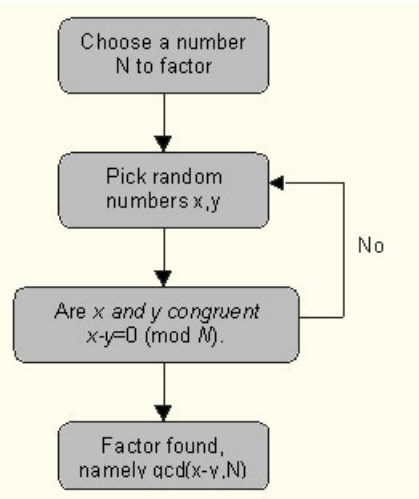
בהמשך הפרק נפרט את דרך הפעולה של אלגוריתם פולרד-רו לפירוק לגורמים וכן את שני אלגוריתמי הנפה, QS ו-NFS.

<u>אלגוריתם פולרד-רו (Pollard's Rho)</u>

למרות שאלגוריתם זה אינו היעיל ביותר כיום, בחרתי להזכיר אותו בעבודה זאת בשל הפופולריות שלו הנובעת מפשטותו היחסית. הוא פותח בסוף שנות השבעים על ידי ג'ורג' מ.פולרד, והוא מסתמך על היוריסטיקה המתבססת על "פרדוקס ימי ההולדת" אשר לפיו מספיקים 23 אנשים בכדי שההסתברות שלשניים מהם יהיה יום הולדת באותו יום גדולה מחצי.

באלגוריתם זה מחפשים אחר שני מספרים בעלי אותה תכונה. אנו מייצרים מספרים אקראיים x ו-y $(x \neq y)$, ומחפשים כאלה שהם קונגרואנטים (מודולו N). כלומר מקיימים $x - y \equiv 0 \pmod N$; $\gcd(x-y, N)$ הוא בהכרח גורם לא טריוויאלי של N. מכיוון שיצירת מספרים ראשוניים היא פעולה אשר צורכת זמן, יוצרים מספרים נוספים על ידי איטרציה של המספרים האקראיים הראשונים תוך שימוש בפולינום בלתי פריק, למשל, $f(x) = x^2 - c$ $(c \neq 2$ כאשר$)$. כך מקבלים סדרת מספרים $x_0 \in Z,\ x_{i+1} = f(x_i)$. סדרת המספרים הזאת יוצרת שובל (tail) של מספרים אשר לאחריה נוצר מחזור (cycle), מעגל סגור אינסופי (צורה של ρ).

נשים לב כי Pollard's Rho לעולם אינו מדפיס תשובה שגויה; כל מספר שהוא מדפיס הוא מחלק לא טריוויאלי של N. אלא ש-Pollard's Rho עשוי שלא להדפיס דבר; אין שום ערובה לכך שהוא יפיק תוצאות כלשהן. אבל קיימות סיבות טובות לצפות שהאלגוריתם ידפיס גורם p של N לאחר $\sqrt{p}$ איטרציות[10]. נספח ב' מציג מימוש של האלגוריתם.



[10] לפירוט, ראה [3] פרק 33.

<u>אלגוריתם הנפה הריבועית (Quadratic Sieve)</u>

בהינתן מודולוס של RSA, $N = pq$, האלגוריתם QS [8] מנסה למצוא שני מספרים y ו- z כך ש- $y \neq \pm z \pmod N$ וגם $y^2 = z^2 \pmod N$. כאשר מוצאים זוג מספרים כזה ניתן לפרק לגורמים את N בקלות מכיוון ש- $\gcd(y-z, N)$ הוא או p או q[11]. על מנת למצוא y ו- z כאלו, אנו יוצרים מספר גדול של ערכים $y_1, y_2, \ldots, y_m$, מחשבים עבור כל אחד את $y_i^2 \pmod N$, ומנסים לפרק אותו למכפלת ראשוניים מבסיס של ראשוניים p_i, המורכב מ- k המספרים הראשוניים הקטנים ביותר $p_1 = 2, p_2 = 3, \ldots, p_k$. מספרים $y_i^2 \pmod N$ בעלי פירוק $\prod_{j=1}^{k} p_j^{e_j}$ ($p_j \in B$) נקראים **חלקים** (smooth). אם מספר הריבועים המודולריים החלקים גדול מ- k, ניתן להשתמש באלימינציה של גאוס (Gauss elimination) כדי למצוא **תת-קבוצה** של וקטורים $(e_1, e_2, \ldots, e_k)$, של ריבויי (multiplicities) הראשוניים שהם תלויים לינארית מודולו 2. כאשר $y_i^2 \pmod N$ המתאימים והפירוק שלהם מוכפלים, מקבלים את המשוואה $\prod_{i=1}^{m} (y_i^2)^{b_i} = \prod_{j=1}^{k} p_j^{c_j} \pmod N$, כאשר כל b_i (אשר מגדירים את תת הקבוצה) הם 0 או 1 וכל c_j (סכומי הריבויים של המספרים הראשוניים) הם מספרים זוגיים. ניתן לקבל, כעת, את המשוואה הרצויה $y^2 = z^2 \pmod N$ על ידי כך שנגדיר $y = \prod_{j=1}^{m} y_i^{b_i} \pmod N$ וגם $z = \prod_{j=1}^{k} p_j^{c_j/2} \pmod N$.

מימוש האלגוריתם:

המפתח ליעילות של QS נעוץ ביצירת מספר גדול של ריבועים מודולריים קטנים אשר ניתן לבדוק האם הם חלקים בצורה קלה. זה מתבצע על ידי שימוש בפולינום הריבועי $f(x) = (a+x)^2 \pmod N$ כאשר $a = \lfloor \sqrt{N} \rfloor$, ובחירת $y_i = a + i$ עבור $i = 1, 2, \ldots, m$. ניתן להראות בקלות שעבור m קטן, $y_i^2 = f(i) \pmod N$ הם ריבועים מודולריים חלקים בסבירות גדולה יותר מאשר ריבועים מודולריים שנבחרו באקראי.

אנו מחזיקים מערך $A[1 \ldots m]$ עבור כל y_i, המאותחל (initialized) לאפס. עבור כל ראשוני p_j בבסיס, פותרים את המשוואה $(a+i)^2 - N \equiv 0 \pmod{p_j}$. נקבל אפס או שני פתרונות d_j, d'_j מודולו p_j. במקרה הראשון, אף $f(i)$ לא יתחלק ב- p_j. במקרה השני, $p_j \mid f(i) \Leftrightarrow i \in \{ p_j r + d_j : r \in Z \} \cup \{ p_j r + d'_j : r \in Z \}$. מגדילים את $A[i]$, לכל i הללו,

[11] $y^2 \equiv z^2 \pmod N \Rightarrow \exists k \in Z,\ y^2 - z^2 = kN \Rightarrow \exists k \in Z, (y+z)(y-z) = kN$

ב- $\log_2 p_j$. מבצעים את זה עבור כל הראשוניים. כעת, סורקים את המערך A , ומחפשים את כל i כך ש- $A[i]$ קרוב למספר הסיביות ב- $f(i)$. בודקים האם $f(i)$ הללו הם חלקים על ידי ניסיונות חלוקה. חוזרים על הפעולה עד אשר נמצאים מספיק ריבועים מודולריים.

דוגמא מספרית:

נניח ש- $N = 4633$ כדוגמא. נתחיל מבחירת קבוצה "קטנה" של ראשוניים, $\{2,3,5\}$. כעת, עבור כל שלם t "קרוב" ל- $\sqrt{N}$, ננסה לפרק לגורמים את $t^2 - N$ תוך שימוש בראשוניים מתוך קבוצת הבסיס הקטנה שבחרנו. פעולה זאת אמורה להיות יעילה הרבה יותר מפירוק לגורמים של N הגדול יותר. אם זה בלתי אפשרי, נעבור ל- t הבא בתור. בדוגמא שלנו $\sqrt{N}$ הוא בקירוב 68.07, ננסה ערכי t בין 38 ל-98, ונקבל (מודולו 4633):

$59^2 \equiv -1152 \equiv -2^7 \cdot 3^2 \cdot 5^0$	(1)	$69^2 \equiv 128 \equiv +2^7 \cdot 3^0 \cdot 5^0$	(4)
$67^2 \equiv -144 \equiv -2^4 \cdot 3^2 \cdot 5^0$	(2)	$85^2 \equiv 2592 \equiv +2^5 \cdot 3^4 \cdot 5^0$	(5)
$68^2 \equiv -9 \equiv -2^0 \cdot 3^2 \cdot 5^0$	(3)	$96^2 \equiv -50 \equiv -2^1 \cdot 3^0 \cdot 5^2$	(6)

חלק זה של האלגוריתם יכול להתבצע במקביל על מחשבים שונים.

כעת, עלינו לבחור מספר משוואות כנ"ל, כך שבהכפלת האגף הימני כל המעריכים יהיו בחזקה זוגית. נבחר במשוואות 3,4,6 ונקבל $68^2 \cdot 69^2 \cdot 96^2 \equiv (-1)^2 \cdot 2^8 \cdot 3^2 \cdot 5^2$. אגף שמאל הוא $(68 \cdot 69 \cdot 96)^2$ והאגף הימני הוא $(2^4 \cdot 3 \cdot 5)^2$. מכאן מקבלים את $x = 68 \cdot 69 \cdot 96$ ו- $y = 2^4 \cdot 3 \cdot 5$ כך ש- $x^2 \equiv y^2 \pmod N$. $x+y$ ו- $x-y$ הם 450672 ו-450192 בהתאמה. $\gcd(x+y, N) = 41$, ולכן 41 הוא גורם משותף של $x+y$ ו- N , ומכאן שהוא גורם לא טריוויאלי של N . לסיכום, מקבלים $4633 = 41 \cdot 113$.

אם כן, אלגוריתם QS מורכב משלושה שלבים:

1) מצא בסיס של מספרים ראשוניים, ופתור את הקונגרואנציות (congruences) $x^2 \equiv N \pmod p$ עבור כל ראשוני בבסיס.
2) ביצוע פעולת **ניפוי** למציאת כמות מספיקה של מספרים חלקים (מספרים $f(x) = (a+x)^2 \pmod N$ כאלה אשר ניתן לפרק לגורמים תוך שימוש בראשונים מהבסיס בלבד).
3) שימוש באלימינציה של גאוס (או שיטה דומה) למציאת מכפלה של מספרים חלקים שהיא ריבוע שלם.

תכונה חשובה של האלגוריתם היא האפשרות לכתוב אלגוריתמים מקביליים אשר ממשים אותו. בפרק התשיעי נתאר התקן חדש (TWINKLE) אשר מנצל את התכונה הזאת של

האלגוריתם ומבצע ניפוי אופטי מהיר ביותר, המאיץ את פעולת האלגוריתם בשלושה סדרי גודל (אבל לא באופן אסימפטוטי).

<u>אלגוריתם נפת שדה המספרים (Number Field Sieve)</u>

אלגוריתם נפת שדה המספרים [8] מבוסס על רעיון מאת ג'ון פולארד (1988), של שימוש בשדות אלגבריים לצורך פירוק לגורמים של מספרים גדולים. בתחילה האלגוריתם יועד לשימוש עבור מספרי פרמה (Fermat numbers)[12], אולם במהרה התגלה כי ניתן להשתמש באלגוריתם עבור מספרים כלשהם.

יהי $f(x)$ פולינום בלתי פריק מעל Z, ו- m מספר שלם כך ש- $f(m) \equiv 0 \mod N$. המעלה d של הפולינום צריכה להיות בערך 6. יהי α שורש מרוכב של f, ונתבונן בחוג (ring) $Z[\alpha]$. מכיוון ש- $f(\alpha) = 0$ ו- $f(m) \equiv 0 \mod N$, על ידי החלפה $m \mod N$ מקבלים מיפוי טבעי ϕ מ- $Z[\alpha]$ ל- Z_N.

<u>משפט 4.2</u>

המיפוי $\phi : Z[\alpha] \to Z_N$ הוא הומומורפיזם של חוגים (ring homomorphism).

הוכחה. זה נובע מיידית מ- $f(\alpha) = 0$ ו- $f(m) \equiv 0 \mod N$.

♦

המטרה היא לבנות קבוצה S של זוגות שלמים זרים $\{(a,b)\}$ כך ש-

$$\prod_{(a,b)\in S}(a - \alpha b) = \gamma^2$$

$$\prod_{(a,b)\in S}(a - mb) = \nu^2$$

יהי $u \equiv \phi(\gamma) \pmod N$. על ידי חישוב מודולו N, מקבלים,

$$\begin{aligned} u^2 &\equiv \phi(\gamma)^2 = \phi(\gamma^2) \\ &= \phi(\prod_{(a,b)\in S}(a - \alpha b)) \\ &= \prod_{(a,b)\in S}\phi(a - \alpha b) \\ &= \prod_{(a,b)\in S}(a - mb) \equiv \nu^2 \pmod N \end{aligned}$$

קיבלנו את המספרים u, v אותם רצינו ליצור (כמו ב-QS). נותר להראות כיצד לבנות את S, f ו- m. נקבע את המעלה d של f ונקבע $m = \left\lceil n^{1/d} \right\rceil$. נרשום את n בבסיס m, כך

[12] מספרי פרמה F_n מוגדרים על ידי $F_n = 2^{2^n} + 1$.

$n = m^d + c_{d-1}m^{d-1} + \ldots + c_1 x + c_0$. הפולינום f , יהיה $f(x) = x^d + c_{c-1}x^{d-1} + \ldots + c_1 x + c_0$.
אם הפולינום בלתי פריק, אנו מקבלים פירוק לגורמים של n , וניתן להראות שהוא לא טריוויאלי. לפרטים נוספים ראה [8].

<u>הישגים בפירוק לגורמים של מספרים גדולים</u>

כאמור, NFS נחשב לאלגוריתם הטוב ביותר כיום לפירוק לגורמים, ולכן משתמשים בו לשבירת שיאים בפירוק לגורמים. קיימת תחרות מתמשכת בין קבוצות שונות ברחבי העולם במטרה לקבוע שיאים חדשים בפירוק לגורמים של מספרים גדולים ביותר. כאשר נקבע שיא חדש, בדרך כלשהי, מספר הספרות של המפתח (שהוא גודל המודולוס N) הופך להיות האתגר הבא לפיצוח של קריפטוגרפים. הטבלה למטה מציגה את ההתקדמות בפיצוח של מערכת RSA בעשור הקודם. ניתן לראות שהאלגוריתם היעיל ביותר כיום הוא NFS.

מספר הספרות	גודל בסיביות	תאריך	האלגוריתם בו השתמשו
RSA-100	330	אפריל 1991	QS
RSA-110	364	אפריל 1992	QS
RSA-120	397	יוני 1993	QS
RSA-129	425	אפריל 1994	QS
RSA-130	430	אפריל 1996	NFS
RSA-140	463	פברואר 1999	NFS
RSA-155	512	אוגוסט 1999	NFS

טבלה 2

על מנת לפתור את RSA-140 (כלומר פירוק לגורמים של מספר בן 140 ספרות או 465 סיביות), נדרשו 200 מחשבים אשר פעלו במקביל במשך ארבעה שבועות כדי לבצע את שלב הניפוי. לאחר מכן, נדרשו למחשב CRAY גדול כ-100 שעות וזיכרון של 810 MB כדי לפתור את מערכת המשוואות. בסיס הראשוניים הכיל 1,500,000 ראשוניים אשר יצרו 4,700,000 משוואות לפתרון. לבעיית RSA-129 אשר נפתרה ב-1994 על ידי שימוש באלגוריתם QS, נדרשו כ-6000 מחשבים שעבדו במקביל (על ידי ניצול של רשת האינטרנט).

כאשר פורסם אלגוריתם RSA לראשונה בסוף שנות ה-70, פרסמו ממציאיו מספר מספרים ב-Scientific American בגדלים שונים. בין המספרים הללו היה RSA-129 אשר פוצח ב-1994. אחד הממציאים (רון ריווסט) טען אז, כי הפירוק לגורמים של מספר זה ימשך כ-40 קוואדריליוני (quadrillion) שנים בטכנולוגיה דאז.

<u>בחירת גודל מפתח</u>

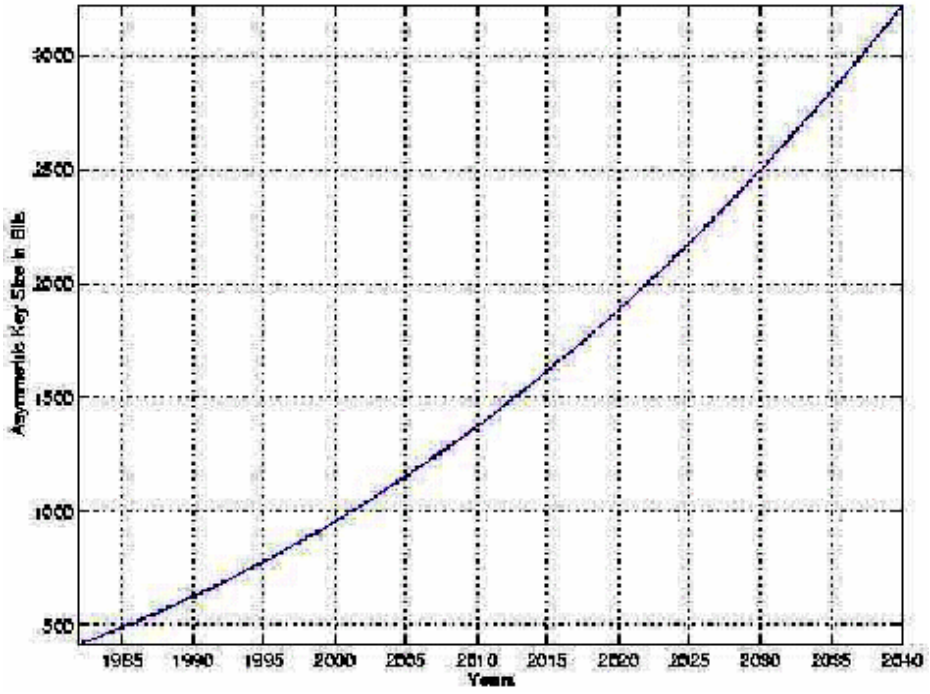
גודל המפתח הנפוץ ביותר היום הוא באורך של 512 סיביות. פירוק לגורמים של מפתח בגודל כזה נעשה לראשונה באוגוסט 1999 והוא בהחלט אפשרי כיום. נשאלת השאלה מהו גודל המפתח הדרוש כדי להבטיח את המידע המוצפן בעתיד הנראה לעין [9]?

טבלה 3 מתארת את הזמן הדרוש לפירוק לגורמים של מפתחות בגדלים שונים, כאשר מתייחסים לזמן הריצה הדרוש לפיצוח מפתח בגודל 465 סיביות כאל יחידת זמן בסיסית (מניחים כי הזמן הדרוש היום לפירוקו הוא 33 ימים).

גודל בסיביות	הזמן הדרוש לניפוי	הזמן הדרוש לפתרון המשוואות	זמן הריצה הדרוש כיום
465	**יחידת זמן 1**	**יחידת זמן 1**	**33 ימים (29+4)**
512	**6**	**9**	**210**
768	**36,000**	**216,000**	1.9×10^6
1012	**40,000,000**	**585,000,000**	3.5×10^9

טבלה 3

ואולם, יש להביא בחשבון גם את ההתפתחות במהירות המחשבים בעתיד (חוק Moore) ואת ההתפתחות האפשרית באלגוריתמים של פירוק לגורמים בבחירת גודל מפתח. מעריכים כי פירוק לגורמים של מספר בן 768 סיביות יהיה אפשרי בקלות יחסית בעוד כ-15 שנה בערך. כיום, ההמלצה היא להשתמש במפתח בגודל 768 סיביות לשימוש אישי, 1024 סיביות לשימוש מסחרי ו-2048 סיביות עבור מידע רגיש במיוחד. האיור למטה מתאר את גודל המפתח המומלץ עבור מערכות מפתח פומבי קלאסיות.



5. התקפות אלמנטריות

בפרק זה אתאר מספר התקפות אלמנטריות על RSA [10]. התקפות אלו לא מצביעות על חולשה של מערכת ההצפנה (האלגוריתם) אלא על שימוש לא נכון בה.

ראשית, אצטט משפט (ללא הוכחה) המציג התקפה על מערכת RSA [2]. הוכחת המשפט מתבססת על תכונת הכפליות של מערכת RSA.

משפט 5.1

אם בידי יריב מצויה שגרה המפענחת ביעילות אחוז אחד של הודעות שנבחרו באופן אקראי מ- Z_N והוצפנו על ידי P_A, אזי הוא יכול להפעיל אלגוריתם הסתברותי אשר בהסתברות גבוהה יפענח כל הודעה שהוצפנה על ידי P_A.

♦

מודולוס משותף (Common Modulus)

על מנת להימנע מהצורך ליצור מודולוס $N = pq$ שונה עבור כל משתמש, ניתן להשתמש במודולוס קבוע לכל המשתמשים. ואולם, שיטה זו מאפשרת לכל משתמש לחשוף את המפתח הסודי של המשתמשים האחרים בדרך הבאה: על פי משפט 4.1, המשתמש מסוגל לפרק את המודולוס N לגורמיו הראשוניים על ידי שימוש במפתח הפרטי והפומבי שלו. ברגע שהוא יודע את הפירוק לגורמים, הוא יכול למצוא את המפתח הסודי של כל משתמש מתוך המפתח הפומבי שלו.

מסקנה: אין להשתמש במודולוס של RSA עבור יותר ממשתמש אחד.

סינוור (Blinding)

יהי (N,d) המפתח הפרטי של בועז ויהי (N,e) המפתח הפומבי שלו. נניח שמארווין רוצה את החתימה של בועז על הודעה $M \in Z_N$. בועז, כמובן, מסרב לחתום על ההודעה M. מארווין יכול לנסות את השיטה הבאה: הוא בוחר באופן אקראי מספר $r \in Z_N$ ומגדיר $M' = r^e M \mod N$. כעת הוא מבקש מבועז לחתום על ההודעה האקראית M'. בועז עשוי להסכים לתת את חתימתו S' על ההודעה התמימה למראה M'. ואולם, נזכור ש- $S' = (M')^d \mod N$. מארווין מחשב $S = S'/r \mod N$ ומקבל את החתימה של בועז S על ההודעה המקורית M. ואכן,

$$S^e \equiv (S')^e / r^e \equiv (M')^{ed} / r^e \equiv M'/r^e \equiv M \mod N$$

.

טכניקה זאת, הקרויה **סינוור** (blinding), מאפשרת למארווין לקבל הודעה חתומה בחתימתו של בועז, זאת על ידי בקשה מבועז לחתום על הודעה "אקראית" לכאורה.

באופן דומה ניתן לבצע התקפה מסוג הודעה מקורית נבחרת (chosen plaintext attack). נניח שמארווין רוצה לפענח הודעה C שהוצפנה תוך שימוש במפתח הציבורי של בועז. מתמטית, מארווין מעוניין בהודעה M, המקיימת,

$$M = C^d \mod N .$$

על מנת למצוא את M הוא בוחר מספר אקראי r קטן מ- N. הוא משתמש במפתח הציבורי של בועז ומחשב,

$$x = r^e \mod N$$
$$y = x^C \mod N .$$
$$t = r^{-1} \mod N$$

כמובן מתקיים, $r = x^d \mod N$.

כעת, מארווין משכנע את בועז לחתום על y עם המפתח הסודי שלו (כזכור, בועז לא ראה מעולם את ההודעה y). כלומר, בועז שולח למארווין,

$$u = y^d \mod N .$$

כדי לקבל את M מארווין מחשב,

$$tu \mod N = r^{-1} \mod N = r^{-1}x^dC^d \mod N = C^d \mod N = M .$$

מסקנה: אין להשתמש ב-RSA כדי לחתום על הודעה אשר מוצגת לך על ידי אדם זר.

6. מעריך פרטי נמוך

ניתן להקטין את הזמן הדרוש לפענוח הודעה (או חתימה על הודעה) במערכת RSA על ידי שימוש בערך קטן של המפתח הסודי d. מכיוון שפעולת העלאה בחזקה מודולרית (modular exponentiation) היא בעלת סיבוכיות זמן ליניארית ב- $\log_2 d$, שימוש במפתח פרטי קטן מאפשר לשפר את ביצועי המערכת פי 10 לפחות (עבור מודולוס בגודל 1024 סיביות). המשפט הבא מתאר התקפה מאת מ.ווינר אשר לפיה שימוש במפתח פרטי קטן מאפשר פיצוח שלם של מערכת ההצפנה [10,11].

משפט 6.1 (מ. ווינר)

יהי $N = pq$ כאשר $q < p < 2q$ ויהי $d < \frac{1}{3} N^{1/4}$. בהינתן (N, e), המקיים $ed = 1 \mod \varphi(N)$, מארווין יכול לחשב באופן יעיל את המפתח הסודי d.

הוכחה. ההוכחה מסתמכת על קירוב באמצעות שברים משולבים (ראה נספח ג').

מכך ש- $ed = 1 \mod \varphi(N)$ נובע כי קיים k כך ש- $ed - k\varphi(N) = 1$. לכן,

$$\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| = \frac{1}{d\varphi(N)}.$$

לפיכך, $\frac{k}{d}$ הוא קירוב של $\frac{e}{\varphi(N)}$. למרות שמארווין אינו יודע את $\varphi(N)$, הוא יכול להשתמש ב- N כדי לקרב אותו. ואכן, מכך ש- $\varphi(N) = N - p - q + 1$[13] וגם $p + q - 1 < 3\sqrt{N}$[14], מקבלים $\left| N - \varphi(N) \right| < 3\sqrt{N}$[15]. על ידי שימוש ב- N במקום $\varphi(N)$, נקבל

$$\left| \frac{e}{N} - \frac{k}{d} \right| = \left| \frac{ed - k\varphi(N) - kN + k\varphi(N)}{Nd} \right| = \left| \frac{1 - k(N - \varphi(N))}{Nd} \right| \le \left| \frac{3k\sqrt{N}}{Nd} \right| = \frac{3k}{d\sqrt{N}}.$$

כעת, $k\varphi(N) = ed - 1 < ed$. מכך ש- $e < \varphi(N)$, רואים ש- $k < d < \frac{1}{3} N^{1/4}$. לכן אנו מקבלים,

$$\left| \frac{e}{N} - \frac{k}{d} \right| \le \frac{1}{d\sqrt[4]{N}} < \frac{1}{2d^2}$$ (יחס קירוב).

מספר השברים $\frac{k}{d}$ ($d < N$) המקרבים את $\frac{e}{N}$ בצורה הזאת, חסום על ידי $\log_2 N$.

---

[13] $\varphi(N) = (p-1)(q-1) = pq - p - q + 1 = N - p - q + 1$.

[14] $p + q - 1 < p + 2p - 1 < 3p < 3\sqrt{N}$.

[15] $\varphi(N) = N - (p + q - 1) > N - 3\sqrt{N} \Rightarrow \left| \varphi(N) - N \right| < 3\sqrt{N}$

למעשה, כל השברים הללו מתקבלים **כמתכנסים** של הפיתוח של $\frac{e}{N}$ לשברים משולבים. כל שצריך לעשות הוא לחשב את $\log_2 N$ המתכנסים של הפיתוח לשברים משולבים של $\frac{e}{N}$. אחד מהמתכנסים הללו ישווה ל-$\frac{k}{d}$ (מהגדרת k ברור כי השבר מצומצם). על ידי מניית המתכנסים כמתואר בנספח ג', ניתן לבצע בדיקה על כל אחד ולגלות בכך את המפתח הסודי d.

♦

לאחרונה (1998) הוכח כי החסם של ווינר אינו הדוק וכל עוד $d < N^{0.292}$, יריב יכול לשחזר את d באופן יעיל מהמפתח הפומבי. משערים כי החסם הנכון הוא $d < N^{0.5}$, ואולם השערה זו לא הוכחה עד לזמן כתיבת עבודה זו.

<u>בעיה פתוחה מספר 2</u>

יהי $N = pq$ ו-$d < N^{0.5}$. בהינתן (N,e) המקיים $ed = 1 \bmod \varphi(N)$ וגם $e < \varphi(N)$, האם ניתן לחשב את d בצורה יעילה?

♦

<u>מניעת ההתקפה</u>

ממשפט 6.1 נובע כי אם המודולוס N מקבל ערך טיפוסי בגודל 1024 סיביות, המפתח הסודי d חייב להיות ארוך מ-256 סיביות בכדי למנוע את ההתקפה. הדבר מצער בעיקר בהתקנים נמוכי הספק כגון כרטיסים חכמים (smart cards) אשר השימוש ב-d נמוך בהם היה חוסך זמן רב. אף על פי כן, ניתן לבצע פענוח מהיר ובכל זאת להימנע מההתקפה הנ"ל. נתאר שתי דרכים עיקריות:

1) **גדול:** e ניתן להשתמש בערך גדול יותר של e, נניח $e' = e + t\varphi(N)$ עבור t גדול. במצב זה, הערך של k בהוכחת המשפט יהיה גדול, וההתקפה היא בלתי אפשרית. חישוב פשוט מראה שאם $e' > N^{1.5}$, אז לא משנה עד כמה d קטן ההתקפה של ווינר לא תהיה ישימה. מצד שני, שימוש בערך גדול של e גורם לכך שזמן ההצפנה יתארך.

2) **שימוש במשפט השאריות הסיני (chinese residue theorem):** נבחר מעריך פרטי d כך ש-$d_p \equiv d \pmod{p-1}$ וגם $d_q \equiv d \pmod{q-1}$ הם קטנים, אבל $d \bmod \varphi(N)$ גדול מ-$\frac{1}{3}N^{1/4}$ (ולכן לא ניתן להשתמש במשפט 6.1). כדי לפענח הודעה, אנו מחשבים

$C^d = C^{d_p} \pmod{p}$ וגם $C^d = C^{d_q} \pmod{q}$ ומשתמשים במשפט השאריות הסיני לחישוב ההודעה M.

איננו יודעים האם השיטות הללו בטוחות, כל שאנו יודעים זה שההתקפה של ווינר אינה אפקטיבית כנגדם.

7. מעריך פומבי נמוך

על מנת להקטין את הזמן הדרוש להצפנת הודעה או לבדיקה של חתימה נהוג להשתמש במעריך פומבי נמוך. שלא כמו התקפות המבוססות על מעריך פרטי נמוך, התקפות המבוססות על מעריך פומבי נמוך רחוקות מפריצה שלמה (total break) [12-14].

משפט קופרסמית (Coppersmith)

ההתקפות החזקות ביותר על מערכות RSA בעלות מעריך פומבי נמוך מבוססות על משפט מאת קופרסמית. למשפט זה קיימים יישומים רבים, ורק חלקם יפורטו כאן. הוכחת המשפט מבוססת על האלגוריתם LLL לצמצום בסיס של סריג (lattice basis reduction) שאתאר בהמשך.

משפט 7.1 (קופרסמית)

יהי N מספר שלם ויהי $f \in Z[x]$ פולינום ממעלה d. נגדיר $X = N^{(1/d)-\varepsilon}$ עבור $\varepsilon \geq 0$ כלשהו. אזי, בהינתן (N, f), מארווין יכול למצוא בצורה יעילה את כל השלמים $|x_0| < X$ המקיימים $f(x_0) = 0 \mod N$. זמן הריצה נקבע על פי זמן הריצה של אלגוריתם LLL על סריג מממד $O(w)$ כאשר $w = \min(1/\varepsilon, \log_2 N)$.

♦

המשפט מספק אלגוריתם למציאה יעילה של כל השורשים של f מודולו N שהם קטנים מ- $X = N^{1/d}$. זמן הריצה של האלגוריתם מתקצר עבור ערכים נמוכים יותר של X. כוחו של המשפט בכך שהוא מאפשר למצוא שורשים קטנים של פולינומים מודולו מספר פריק N. כאשר עובדים מודולו מספר ראשוני אין סיבה להשתמש במשפט קופרסמית מכיוון שקיימים אלגוריתמים אחרים יעילים הרבה יותר.

נתאר את הוכחת המשפט בקווים כלליים. בהינתן פולינום $h(x) = \sum a_i x^i \in Z[x]$, נגדיר נורמה $\|h\|^2 = \sum_i |a_i|^2$. ההוכחה מתבססת על הלמה הבאה.

למה 7.2

יהי $h(x) \in Z[x]$ פולינום ממעלה d, ויהי X מספר שלם חיובי. נניח $\|h(X)\| < N/\sqrt{d}$. אם $|x_0| < X$ מקיים $h(x_0) = 0 \bmod N$, אזי $h(x_0) = 0$ מתקיים מעל השלמים.

הוכחה. נשתמש באי-שוויון קושי-שוורץ

$$|h(x_0)| = \left|\sum a_i x_0^i\right| = \left|\sum a_i X^i \left(\frac{x_0}{X}\right)^i\right| \le \sum \left|a_i X^i \left(\frac{x_0}{X}\right)^i\right| \le \sum |a_i X^i| \le \sqrt{d}\,\|h(X)\| < N$$

מכיוון ש- $h(x_0) = 0 \bmod N$, מסיקים ש- $h(x_0) = 0$ לכל מספר שלם.

♦

כלומר, אם h הוא פולינום בעל נורמה קטנה, אזי כל השורשים הקטנים של $h \bmod N$ הם גם שורשים של h מעל השלמים. ניתן להשתמש בלמה לצורך מציאת שורשים קטנים, למשל x_0, של $f(x) \bmod N$, על ידי חיפוש אחר פולינום אחר $h \in Z[x]$ בעל נורמה נמוכה ואותם שורשים כמו של $f \bmod N$. במקרה כזה, x_0 הוא שורש של h מעל השלמים וניתן למצוא אותו בקלות. על מנת לעשות זאת, עלינו לחפש אחר פולינום $g \in Z[x]$ כך של- $h = gf$ יש נורמה קטנה (כלומר קטנה מ- N). כלומר, יש למצוא צירוף ליניארי עם מקדמים שלמים (integer linear combination) של הפולינומים $f, xf, x^2 f, \ldots, x^r f$ בעל נורמה נמוכה. למרבה הצער, לעיתים קרובות לא קיימים צירופים ליניאריים לא טריוויאליים בעלי נורמה מספיק נמוכה. אף על פי כן, ניתן להראות כי על ידי שימוש בפולינומים מהצורה $g_{u,v}(x) = N^{m-v} x^u f(x)^v$, תמיד ניתן למצוא צירוף ליניארי כנ"ל עבור m מספיק גדול[16]. ברגע שמצאנו $h(x)$ כנדרש, מלמה 7.2 נובע כי x_0 הוא שורש מעל השלמים ומכאן שניתן למצוא אותו בקלות.

נותר להראות כיצד למצוא את $h(x)$ בצורה יעילה. לצורך כך, עלינו להשתמש במושג **השריג** מעל Z^w. יהיו $u_1, \ldots, u_w \in Z^w$ וקטורים בלתי תלויים ליניארית. סריג L הנפרש על ידי $\langle u_1, \ldots, u_w \rangle$ הוא קבוצת הצירופים הליניאריים עם מקדמים שלמים של הווקטורים $u_1, \ldots, u_w$. הדטרמיננטה של L מוגדרת כדטרמיננטה של מטריצה ריבועית בגודל $w \times w$, אשר השורות בה הם הווקטורים $u_1, \ldots, u_w$.

במקרה שלנו, נתייחס לפולינומים $g_{u,v}(X)$ כווקטורים ונתבונן בשריג L הנפרש על ידם. קובעים $u = 0, \ldots, d-1$ ו- $v = 0, \ldots, m$, ומכאן שממד הסריג הוא $w = d(m+1)$.

[16] פתרון זה לבעיה מבוסס על כך שאם $f(x_0) = 0 \bmod N$ אזי $f(x_0)^k = 0 \bmod N^k$ לכל k.

לדוגמא, כאשר f הוא פולינום ריבועי ו-$m=3$, השריג המתקבל, נפרש על ידי השורות של המטריצה הריבועית הבאה:

$$\begin{array}{c} \\ g_{0,0}(xX) \\ g_{1,0}(xX) \\ g_{0,1}(xX) \\ g_{1,1}(xX) \\ g_{0,2}(xX) \\ g_{1,2}(xX) \\ g_{0,3}(xX) \\ g_{1,3}(xX) \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{array} \\ \begin{bmatrix} N^3 & & & & & & & \\ & XN^3 & & & & & & \\ * & * & X^2N^2 & & & & & \\ & * & * & X^3N^2 & & & & \\ * & * & * & * & X^4N & & & \\ & * & * & * & * & X^5N & & \\ * & * & * & * & * & * & X^6 & \\ & * & * & * & * & * & * & X^7 \end{bmatrix} \end{array}$$

התאים המסומנים ב-* מתאימים למקדמים של הפולינומים אשר מהם אנו מתעלמים. התאים הריקים הם אפסים. מכיוון שהמטריצה היא משולשת, הדטרמיננטה שלה היא מכפלת האלמנטים באלכסון. מטרתנו היא למצוא ווקטורים קצרים (כלומר, בעלי נורמה קטנה) בסריג זה.

משפט ידוע מאת הרמיט (Hermite) קובע כי כל שריג L בעל ממד w מכיל נקודה $v \in L$ שונה מאפס שהנורמה שלה מקיימת $\|v\| \le \gamma_w \det(L)^{1/w}$, כאשר γ_w הוא קבוע אשר תלוי רק ב-w. ניתן להשתמש במשפט הרמיט ובחסם שהוא מגדיר, כדי להראות שעבור m מספיק גדול השריג שלנו מכיל ווקטורים בעלי נורמה קטנה מ-N^m, כנדרש. השאלה היא האם ניתן למצוא בצורה **יעילה** ווקטור קצר ב-L אשר אורכו לא גדול מהחסם של משפט הרמיט. אלגוריתם LLL, הוא אלגוריתם יעיל אשר מבצע בדיוק את זה.

<u>עובדה 7.3 (LLL)</u>

יהי L הסריג הנפרש על ידי $\langle u_1,\ldots,u_w \rangle$, כאשר $\langle u_1,\ldots,u_w \rangle$ נתונים כקלט. אזי האלגוריתם LLL מחזיר כפלט נקודה $v \in L$ המקיימת $\|v\| < 2^{w/4} \det(L)^{1/w}$. זמן הריצה של LLL הוא ריבועי באורך של הקלט.

♦

לאלגוריתם LLL (על שם ממציאיו L.Lovasz, A.Lenstra & H.Lenstra) קיימים יישומים רבים בתורת המספרים החישובית (computational number theory) ובקריפטוגרפיה. הוא התגלה בשנת 1982 ומאפשר לפרק לגורמים פולינומים מעל השלמים בצורה יעילה (ובאופן כללי מעל חוגי מספרים). מערכות הצפנה רבות נפרצו על ידי שימוש באלגוריתם LLL (בפרט, מערכות הצפנה המבוססות על בעיית תרמיל הגב, "Knapsack problem").

שימוש באלגוריתם LLL מאפשר לנו להשלים את הוכחת משפט קופרסמית. כדי להבטיח שהוקטור שנוצר על ידי LLL מקיים את החסם של למה 7.2 עליו לקיים,

$$2^{w/4}\det(L)^{1/w} < N^m/\sqrt{w}$$

כאשר $w=d(m+1)$ הוא הממד של L. חישוב שגרתי מראה כי עבור m גדול מספיק, התנאי מתקיים. אכן, אם $X = N^{\frac{1}{d}-\varepsilon}$, מספיק לקחת $m=O(k/d)$ כאשר $k=\min(\frac{1}{\varepsilon},\log N)$. כתוצאה מכך זמן הריצה נקבע על ידי זמן הריצה של אלגוריתם LLL על שריג שגודלו $O(k)$, כנדרש.

נציג כעת מספר התקפות אשר מתבססות על משפט קופרסמית.

<u>התקפה של האסטד (Hastad's Broadcast Attack)</u>

נניח שבועז רוצה לשלוח הודעה מוצפנת, M, למספר צדדים, $P_1, P_2, \ldots, P_k$. לכל צד יש מפתח פומבי (N_i, e_i) משלו. כמוכן נניח ש- M קטנה יותר מכל N_i. בועז שולח את ההודעה המוצפנת לכל P_i (תוך שימוש במפתח הפומבי המתאים). נניח שמארווין מצותת ומשיג את k ההודעות המוצפנות.

למען הפשטות נניח כי כל המעריכים הציבוריים, e_i, שווים ל-3. נראה כעת שמארווין יכול לשחזר את ההודעה המקורית M אם $k \geq 3$. מארווין מקבל את C_1, C_2, C_3, המקיימים

$$C_1 = M^3 \mod N_1$$

$$C_2 = M^3 \mod N_2$$

$$C_3 = M^3 \mod N_3$$

ניתן להניח שמתקיים $\gcd(N_i, N_j) = 1$ עבור כל $i \neq j$[17]. נשתמש במשפט השאריות הסיני עבור C_1, C_2, C_3 ונקבל $C' \in Z_{N_1N_2N_3}$ המקיים $C' = M^3 \mod N_1N_2N_3$. מכיוון ש- M קטן מכל N_i, מתקיים $M^3 < N_1N_2N_3$. אזי $C' = M^3$ מתקיים מעל השלמים. לפיכך, מארווין יכול למצוא את M על ידי חישוב השורש המעוקב של C'. באופן כללי, אם כל המעריכים הפומביים שווים לערך e, מארווין יכול לשחזר את M כאשר $k \geq e$. ההתקפה אפשרית, אם כן, רק כאשר משתמשים בערך קטן של e.

האסטד מתאר התקפה חזקה הרבה יותר. נניח שעבור כל אחד מהצדדים, $P_1, \ldots, P_k$, בועז מחזיק פולינום פומבי קבוע $f_i \in Z_{N_i}[x]$. כדי לשדר הודעה M, בועז שולח את ההצפנה של $f_i(M)$

[17] אחרת, מארווין יכול לפרק לגורמים חלק מ- N_i.

לצד P_i[18]. בצורה כזאת בועז שואף למנוע את ההתקפה אשר תוארה בפסקה הקודמת. למרבה הצער, האסטד הראה כי שימוש בפולינום בצורה שתוארה אינו מונע את ההתקפה אם מספיק צדדים משתתפים. על ידי ציטוט, מארווין לומד כי $C_i = f_i(M)^{e_i} \mod N_i$ עבור $i = 1,\ldots,k$. המשפט הבא מתאר כיצד יכול מארווין לשחזר את ההודעה M אם מספיק צדדים משתתפים.

משפט 7.4 (Hastad)

יהיו $N_1,\ldots,N_k$ מספרים שלמים זרים אחד לשני, ונגדיר $N_{\min} = \min_i(N_i)$. יהיו $g_i \in Z_{N_i}[x]$, k פולינומים בעלי מעלה מרבית d. נניח שקיים $M < N_{\min}$ יחיד המקיים $g_i(M) = 0 \mod N_i$ עבור כל $i = 1,\ldots,k$. תחת ההנחה ש-$k > d$, ניתן למצוא בצורה יעילה את M בהינתן $(N_i, g_i)_{i=1}^k$.

הוכחה. יהי $\overline{N} = N_1 \cdots N_2$. על ידי מכפלת כל g_i בחזקה מתאימה של x ניתן להניח שלכולם יש מעלה d. נבנה את הפולינום

$$g(x) = \sum_{i=1}^{k} T_i g_i(x) \quad \text{כאשר} \quad T_i = \begin{cases} 1 \mod N_j & i = j \\ 0 \mod N_j & i \neq j \end{cases}.$$

T_i הם שלמים הידועים כמקדמי השארית הסינית. מעלת $g(x)$ היא d. כמוכן, אנו יודעים ש-$g(M) = 0 \mod \overline{N}$, וגם $M < N_{\min} \le \overline{N}^{1/k} < \overline{N}^{1/d}$ ומכאן על סמך משפט 7.1 משי"ל.

♦

המשפט מראה כי ניתן לפתור בצורה יעילה מערכת של משוואות של משתנה אחד מודולו מספרים זרים, בהנחה שנתונות מספיק משוואות. על ידי ההצבה $g_i = f_i^{e_i} - C_i \mod N_i$, מארווין יכול לשחזר את M מהצפנים הנתונים כל אימת שמספרם של הצפנים הוא לפחות d, כאשר d הוא המקסימום בין $e_i \deg(f_i)$ עבור כל $i = 1,\ldots,k$. בפרט, אם כל e_i שווים ל-e ובועז שולח הודעות קשורות ליניארית, מארווין יכול לשחזר את ההודעה כאשר $k > e$.

מכאן, שעל מנת להתגונן כהלכה בפני סוג ההתקפות הזה יש לרפד את ההודעה באופן אקראי ולא בצורה קבועה.

התקפה של פרנקלין-ריטר (Franklin-Reiter Related Message Attack)

התקפה זאת מתייחסת למצב אשר בו בועז שולח לאיה הודעות מוצפנות (באופן שיתואר בהמשך) המשתמשות באותו מודולוס. יהי (N,e) המפתח הפומבי של איה. נניח $M_1, M_2 \in Z_N^*$ הן שתי

[18] לדוגמא, אם ההודעה M היא באורך m סיביות, בועז יכול לשלוח לצד i את ההודעה $M_i = i2^m + M$.

הודעות שונות המקיימות $M_1 = f(M_2) \bmod N$ עבור פולינום ידוע מסוים $f \in Z_N[x]$. כדי לשלוח את M_1 ואת M_2 לאיה, בועז מצפין את ההודעות ושולח את הצפנים המתאימים C_1, C_2. אנו נראה, כי בהינתן C_1, C_2, מארווין יכול לשחזר בצורה יעילה את M_1 ואת M_2. אל אף שההתקפה עובדת עבור כל e נמוך, אנו ננסח את הטענה עבור $e = 3$, ללא הוכחה.

משפט 7.5 (FR)

נניח $e = 3$, ויהי (N, e) מפתח RSA פומבי. יהי $M_1 \neq M_2 \in Z_N^*$ מקיימים $M_1 = f(M_2) \bmod N$ עבור פולינום ליניארי $f = ax + b \in Z_N[x]$ ($b \neq 0$). אזי, בהינתן (N, e, C_1, C_2, f), ניתן לשחזר את M_1, M_2 בזמן ריבועי ב- $\log N$.

♦

עבור $e > 3$ ההתקפה צורכת זמן ריבועי ב- e. לכן, ניתן להשתמש בהתקפה עבור מעריך פומבי קטן בלבד.

התקפה של קופרסמית (Coppersmith's Short Pad Attack)

ההתקפה הבאה מראה את הסכנה שבשימוש בריפוד פשטני של הודעה. נניח שבועז שולח לאיה הודעה מוצפנת M, מרופדת כיאות על ידי הוספת מספר סיביות אקראיות לסוף ההודעה. מארווין מיירט את ההודעה ומונע ממנה מלהגיע ליעדה. בועז רואה שאיה אינה מגיבה להודעה שלו והוא מחליט לשלוח את M מחדש לאיה. הוא מרפד את ההודעה M ושולח את ההודעה המוצפנת. למארווין יש כעת שתי הודעות המתאימות לשתי הצפנות שונות של אותה הודעה המרופדת בצורה אקראית. המשפט הבא (מובא ללא הוכחה) מראה כי למרות שמארווין אינו יודע את הריפוד (pad) בו השתמשו, הוא יכול לשחזר את M ביעילות. הוכחת המשפט מסתמכת על משפט קופרסמית.

משפט 7.6

יהי (N, e) מפתח RSA פומבי, כאשר N הוא בעל אורך של n סיביות. נקבע $m = \lceil n/e^2 \rceil$. תהי $M \in Z_N^*$ הודעה בעלת אורך של לכל היותר $n - m$ סיביות. נגדיר $M_1 = 2^m M + r_1$ ו- $M_2 = 2^m M + r_2$, כאשר r_1 ו- r_2 הם שלמים שונים המקיימים $0 \leq r_1, r_2 < 2^m$. אם ניתן למארווין (N, e) וגם הצפנים C_1, C_2 של M_1, M_2 (אבל אין לו את r_1 או r_2), הוא יכול לשחזר את M בצורה יעילה.

הוכחה: נגדיר $g_1(x, y) = x^e - C_1$ וגם $g_2(x, y) = (x + y)^e - C_2$. אנו יודעים כי כאשר $y = r_2 - r_1$, M_1 הוא שורש משותף של שני הפולינום. במילים אחרות, $\Delta = r_2 - r_1$ הוא שורש של הפולינום $h(y) = res_x(g_1, g_2) \in Z_N[y]$. מעלת h היא לכל היותר e^2. בנוסף, $|\Delta| < 2^m < N^{1/e^2}$. כלומר, Δ הוא שורש "קטן" של h מודולו N, ומארווין יכול למצוא אותו בקלות תוך שימוש

במשפט קופרסמיט (משפט 6.1). ברגע שיודעים את Δ, ניתן להשתמש בהתקפה של FR (סעיף קודם) כדי למצוא את M_2 ואת M.

♦

כאשר $e=3$, ניתן ליישם את ההתקפה כל עוד אורך הריפוד הוא פחות מ- $1/9$ מאורך ההודעה. יש לציין שעבור הערך המקובל $e=65537$, ההתקפה חסרת ערך עבור כל גודל מודולוס תיקני.

<u>התקפה המבוססת על חשיפה חלקית של המפתח (Partial Key Exposure Attack)</u>

יהי (N,d) מפתח RSA פרטי. נניח שבדרך כלשהי מארווין מסוגל לחשוף חלק מהסיביות של d, נאמר רבע מהם. האם ניתן לשחזר את שאר הסיביות של d? באופן מפתיע, התשובה היא חיובית עבור ערכים קטנים של e. מכאן החשיבות בהגנה על המפתח הפרטי *כולו* מפני חשיפה.

<u>משפט 7.7 (BDF)</u>

יהי (N,e) מפתח RSA פרטי כלשהו כאשר N באורך של n סיביות. בהינתן $\lceil n/4 \rceil$ הסיביות הפחות משמעותיות של d, מארווין יכול לשחזר את כל הספרות של d בזמן ליניארי ב- $e\ \log_2 e$.

הוכחה. הוכחת המשפט מתבססת על המשפט הבא מאת קופרסמית,

<u>משפט 7.8 (קופרסמית)</u>

יהי $N=pq$ מודולוס RSA באורך n סיביות. אזי בהינתן $n/4$ הסיביות הפחות משמעותיות של p או $n/4$ הסיביות הכי משמעותיות של p, ניתן לפרק לגורמים בצורה יעילה את N.

מהגדרת e ו- d, קיים מספר שלם k כך ש- $ed-k(N-p-q+1)=1$. מכיוון ש- $d<\varphi(N)$, חייב להתקיים $0<k\le e$. לאחר צמצום המשוואה מודולו $2^{n/4}$ והצבת $q=N/p$, מקבלים $(ed)p-kp(N-p+1)+kN=p \mod 2^{n/4}$. מכיוון שמארווין יודע את $n/4$ הספרות הפחות משמעותיות של d, הוא יודע את הערך של $ed \mod 2^{n/4}$. כתוצאה מכך, מקבלים משוואה ב- k ו- p. עבור כל e הערכים האפשריים של k, מארווין פותר את המשוואה הריבועית ב- p ומקבל מספר ערכים אפשריים של $p \mod 2^{n/4}$. עבור כל אחד מהערכים הללו, מארווין מריץ את האלגוריתם של משפט 6.8 במטרה לפרק לגורמים את N. ניתן להראות כי מספר הערכים של $p \mod 2^{n/4}$ הוא לכל היותר $e\cdot\log_2 e$. מכאן שאחרי לכל היותר $e\cdot\log_2 e$ בדיקות, ניתן לפרק לגורמים את N.

♦

משפט 7.7 ידוע כהתקפה המבוססת על חשיפה חלקית של מפתח (Partial key-exposure attack). התקפות דומות קיימות עבור ערכים גדולים יותר של e, כל עוד $e < \sqrt{N}$.

8. התקפות מימוש

נפנה כעת את תשומת הלב להתקפות מסוג אחר, אשר במקום לתקוף את המבנה הבסיסי של פונקצית ה-RSA, הן מתמקדות במימושים מסוימים של RSA.

התקפות תזמון (Timing Attacks)

נתייחס לכרטיס "חכם" (smart card)[19] אשר אוגר בתוכו מפתח RSA פרטי. מכיוון שהכרטיס חסין בפני פגיעה, כמובן, מארווין לא יכול לבחון את תוכנו ולחשוף בכך את המפתח הסודי. Kocher [15] הראה שעל ידי מדידה מדויקת של **הזמן** הדרוש לביצוע פענוח או חתימה, מארווין יכול לגלות את המפתח הסודי d בצורה מהירה.

נראה כיצד ניתן להשתמש בהתקפה זאת כנגד מימוש פשוט של RSA המשתמש **באלגוריתם העלאה חוזרת בריבוע** (Repeated squaring algorithm). יהי $d = d_n d_{n-1} \ldots d_0$ הייצוג הבינארי של d[20]. האלגוריתם מחשב את $C = M^d \mod N$, על ידי שימוש לכול היותר ב-$2n$ הכפלות מודולריות (Modular multiplication), בהתבסס על ההבחנה ש-$C = \prod_{i=0}^{n} M^{2^i d_i} \mod N$, והוא עובד באופן הבא:

$z \leftarrow M; C \leftarrow 1.$ For $i = 0,1,\ldots,k$ do:

1. If $d_i = 1$, $C \leftarrow Cz \pmod N$.
2. $z \leftarrow z^2 \pmod N$.

על מנת ליישם את ההתקפה, מארווין מבקש מהכרטיס חכם ליצור חתימות על מספר גדול של מסרים אקראיים $M_1, M_2, \ldots, M_n$ ולמדוד את הזמן T_i הנדרש לכרטיס ליצור כל אחת מהחתימות.

ההתקפה משחזרת את הסיביות (bits) של d אחד בכל פעם, החל מהסיבית הפחות משמעותית (least significant bit). אנו יודעים כי d הוא אי זוגי ולכן $d_0 = 1$. כעת, אם $d_1 = 1$ הכרטיס מחשב את המכפלה $Cz = M \cdot M^2$, אחרת הוא לא. יהי t_i הזמן הנדרש לכרטיס לחשב את $M_i \cdot M_i^2 \pmod N$. t_i נבדלים אחד מהשני מכיוון שהזמן הנדרש לחישוב תלוי בערך של M_i[21]. מארווין מודד את הזמנים t_i במצב לא מקוון (off-line), תוך שימוש במפרט של הכרטיס חכם.

[19] כרטיס חכם הינו כרטיס מגנטי המכיל מיני-מעבד וזיכרון ומבצע פעולות ממוחשבות שונות, למשל - כרטיס אשראי, ארנק אלקטרוני וכו'.

[20] כלומר $d = \sum_{i=0}^{n} 2^i d_i$ כאשר $d_i \in \{0,1\}$.

[21] אלגוריתמים לצמצום מודולרי פשוטים פועלים במשך זמן שונה בתלות במספר הנתון.

Kocher הבחין שכאשר $d_1 = 1$, $\{t_i\}$ ו- $\{T_i\}$ מתואמים (correlated)[22]. מצד שני, אם $d_1 = 0$, $\{t_i\}$ ו- $\{T_i\}$ הם בלתי תלויים. על ידי מדידת מידת המתאם מארווין יכול לקבוע האם $d_1 = 1$ או $d_1 = 0$. באופן דומה ניתן לקבוע את הערכים של הסיביות הגבוהות יותר.

<u>מניעת ההתקפה</u>

קיימות שתי שיטות למניעת ההתקפה:

1. השיטה הפשוטה יותר היא להוסיף השהיות מתאימות כך שפעולת ההעלאה בחזקה מודולרית תימשך תמיד פרק זמן זהה.

2. השיטה השניה מבוססת על סינוור (blinding). טרם ביצוע הפענוח של M, הכרטיס חכם מייצר מספר אקראי $r \in Z_N^*$ ומחשב את $M' = M \cdot r^e \mod N$. כעת הוא מבצע את ההצפנה על M' ומקבל $C' = (M')^d \mod N$. בסופו של דבר מחושב $C = C' / r \mod N$. בצורה זאת ההצפנה מתבצעת על מסר שאינו ידוע למתקיף מארווין.

<u>התקפות הספק (Power Cryptanalysis)</u>

Kocher הראה בנוסף, שעל ידי מדידה מדויקת של ההספק הנצרך על ידי הכרטיס חכם במשך תהליך יצירת החתימה, מארווין יכול (בדרך כלל) לגלות בקלות את המפתח הפרטי. התקפה זאת מבוססת על העובדה, שבמשך ביצוע מכפלה בדרגת דיוק גבוהה צריכת ההספק של הכרטיס היא גבוהה מהרגיל. על ידי מדידת אורכם של פרקי הזמן בהם צריכת ההספק גבוהה יותר, מארווין יכול לקבוע בקלות האם במהלך איטרציה מסוימת, הכרטיס מבצע שתי מכפלות או רק אחת, ועל ידי כך לגלות את הסיביות של d.

<u>התקפות המנצלות שגיאות אקראיות (Random Faults)</u>

קיימים מימושים של מערכת ההצפנה RSA אשר משתמשים במשפט השאריות הסיני (chinese remainder theorem) על מנת לזרז את החישוב של $M^d \mod N$ (ראה למשל בפרק 6, "מעריך פרטי נמוך"). במקום לעבוד מודולו N, בועז מחשב את החתימה/מסר מודולו p ו- q ולאחר מכן מאחד את התוצאות על ידי שימוש במשפט השאריות הסיני. ביתר פירוט, בועז מחשב קודם

$$C_p = M^{d_p} \mod p \qquad C_q = M^{d_q} \mod q$$

כאשר $d_p = d \mod (p-1)$ וגם $d_q = d \mod (q-1)$. כעת, הוא מקבל את החתימה C על ידי

[22] למשל, אם עבור i מסוים, t_i גדול בהרבה מערך התוחלת שלו, אזי T_i כנראה יהיה גדול מערך התוחלת שלו גם כן.

$$C = T_1 C_P + T_2 C_q \mod N$$

כאשר,

$$T_1 = \begin{Bmatrix} 1 \mod p \\ 0 \mod q \end{Bmatrix} \quad ; \quad T_2 = \begin{Bmatrix} 0 \mod p \\ 1 \mod q \end{Bmatrix}.$$

זמן הריצה של פעולת החישוב האחרונה קטן באופן משמעותי מזמן הריצה של פעולות ההעלאה בחזקה. נשים לב שאורכם של p ו-q הוא חצי מאורכו של המודולו N. כמוכן, גודלם של d_p ו-d_q הוא חצי מאורכו של d. מכך שפעולת ההעלאה בחזקה המודולרית היא בעלת זמן ריצה ריבועי נובע כי חישוב C_p ו-C_q קצר פי שמונה מחישוב C. כלומר, השימוש בשיטה מצמצם את הזמן הדרוש לחתימה/פענוח פי ארבעה. מסיבה זאת, מימושים רבים משתמשים בשיטה זאת לשיפור הביצועים.

Boneh, Demillo ו-Lipton [16] הראו שקיימת סכנה מובנית בשימוש במשפט השאריות הסיני. נניח כי במהלך יצירת חתימה, תקלה קטנה במחשבו של בועז גורמת לו לבצע חישוב שגוי במהלך אחת מהפעולות. למשל במהלך ההעתקה של אוגר (register) ממקום אחד לאחר, אחת מהסיביות מתהפכת (תקלה מסוג זה עלולה להתרחש, למשל, כתוצאה מהפרעות אלקטרומגנטיות או מתקלה נדירה בחומרה). בהינתן חתימה שגויה, מארווין יכול לפרק לגורמים בקלות את המודולוס של בועז, N.

נניח שמתרחשת שגיאה יחידה במהלך פעולת יצירת החתימה על ידי בועז. כתוצאה מכך C_p או C_q (אחד מהם) יחושב שלא כהלכה. נניח ש-C_p הוא נכון, אבל $\hat{C}_q$ לא. החתימה המתקבלת היא $\hat{C} = T_1 C_p + T_2 \hat{C}_q$. כאשר מארווין מקבל את $\hat{C}$, הוא יודע מיד כי מדובר בחתימה שגויה, מכיוון ש-$\hat{C}^e \neq M \mod N$. ואולם, שים לב כי

$$\hat{C}^e = M \mod p \quad \text{בעוד ש} \quad \hat{C}^e \neq M \mod q$$

כתוצאה מכך $\gcd(N,\ \hat{C}^e - M)$, חושף גורם לא טריוויאלי של המודולוס N.

על מנת שההתקפה תעבוד, מארווין חייב להכיר באופן מלא את ההודעה M. כלומר, אנו מניחים כי בועז לא משתמש בשיטת הסינוור. ריפוד ההודעה בצורה אקראית מונע לחלוטין את האפשרות להשתמש בשיטה הזאת. שיטה פשוטה נוספת למניעת ההתקפה היא בדיקת החתימה לפני פרסומה.

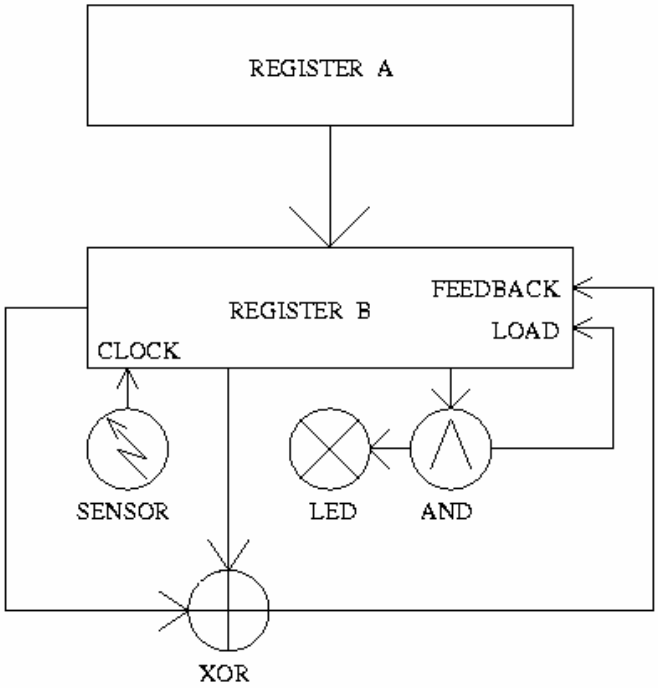
9. המילה האחרונה: TWINKLE

ב-EUROCRYPT'99, תיאר עדי שמיר [17] ממכון ויצמן התקן חישוב אופטי אשר מסוגל להאיץ את קצב החישוב בשלב הניפוי (מציאת מספרים חלקים) של אלגוריתם QS פי 500 עד פי 1000. ההתקן ניקרא TWINKLE[23] ועלות הייצור המשוערת שלו היא כ-5000 דולר. ה-TWINKLE מאפשר להגדיל את טווח המספרים שניתן לפרק ב-200 סיביות (עבור אותה השקעה של זמן וכסף). חשיבות ההתקן היא בכך שהוא יכול לשמש לפריצת מערכות RSA של 512 סיביות, המשמשות כיום בצורה נרחבת ביישומי מסחר אלקטרוני באינטרנט (e-commerce).

לדעתי, ה-TWINKLE הוא דוגמא להישג מדעי ואקדמי מרשים בכך שהוא תוקף בעיה בעלת חשיבות מעשית רבה, ומשתמש לצורך כך במגוון של דיסציפלינות מדעיות החל מתורת המספרים האלגברית וכלה בטכנולוגיות מתקדמות של מוליכים למחצה.

מבנה ה-TWINKLE

התקן ה-TWINKLE מורכב מגליל מושחר, המכיל מערך של לדים (LED, דיודות פולטות אור) בתחתית, ופוטודטקטור (photodetector) בראש. הפוטודטקטור מודד את כמות האור הכללית



הנפלטת על ידי מערך הלדים בכל רגע, ומתריע למחשב PC כאשר יש חריגה מערך מוגדר מסוים. חריגה מסוג זה קשורה לזיהוי של מספרים חלקים.

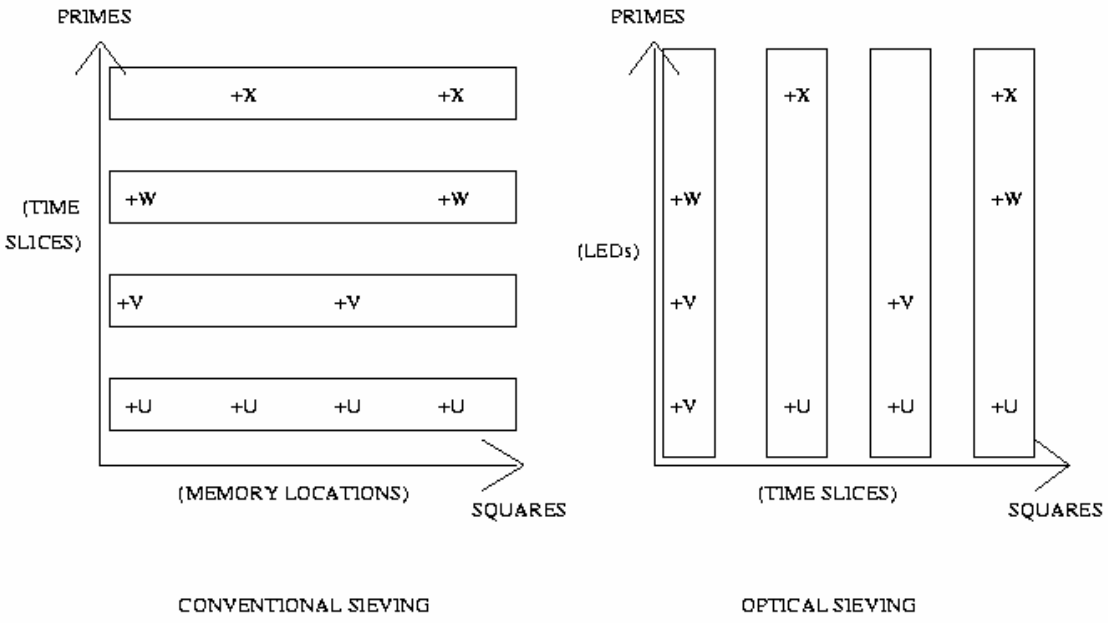
[23] ראשי תיבות של The Weizmann INstitute Key Locating Engine.

כל לד (LED) משויך למשך זמן מסוים p_j ($p_j \in B$) ולהשהיה מסוימת d_j, והוא פשוט מאיר למשך מחזור שעון אחד (clock cycle) בזמנים הנתונים על ידי $p_j r + d_j$, $r \geq 0$. ברצוננו שהלד יפלוט אור בעוצמה (intensity) יחסית ל- $\log_2 p_j$ כאשר הוא זוהר. משיגים זאת על ידי שימוש במערך אחיד של לדים ומסנן אופטי מתאים.

מערך הלדים נבנה על פיסה יחידה של GaAs (גאליום ארסנייד). לכל תא יש שני אוגרים (ראה איור בעמוד הקודם, המתאר מבנה של תא יחיד במערך) אשר מאותחלים עם הערכים המתאימים p_j ו- d_j. בכל תא יש גם פוטודטקטור, אשר מיועד לזהות LED פעיל.

<u>מדוע TWINKLE מהיר יותר?</u>

מימוש רגיל של תהליך הניפוי מציב ריבועים מודולריים (modular squares) בתאים של מטריצה (שימוש במקום) וביצוע לולאה על המספרים הראשוניים (שימוש בזמן). התקן ה-TWINKLE מקצה לדים למספרים ראשוניים (שימוש במקום) ומבצע לולאה על הריבועים המודולריים (שימוש בזמן), ובכך הופך את תפקידם (זמן במקום מקום, ראה איור).



הסיבות לכך שפעולת הניפוי ב-TWINKLE מהירה יותר הן:

1) מחזור שעון מהיר ביותר, 10 Ghz (לעומת 200 Mhz בחומרה רגילה). זה נובע מכך שטכנולוגיית GaAs יכולה לתמוך במהירויות שעון גדולות יותר מאשר שבבי RAM.

2) ניתן לחבר בו זמנית 200,000 מספרים ב-TWINKLE באמצעות הפוטודטקטור. בניית מחבר ספרתי אשר מחבר כל כך הרבה מספרים במחזור שעון אחד היא לגמרי לא מציאותית.

3) אין צורך לסרוק מערכים גדולים של מונים, ואין צורך לבצע אתחול.

יש לציין, כי ההתקן החדש משפר את מהירות החלק הראשון (שלב הניפוי) בלבד של אלגוריתם QS והוא שווה ערך ל-100 עד 1000 מחשבים רגילים. ההתקן אינו משפר את מהירות השלב השני של האלגוריתם (פתרון המשוואות המודולריות).

<u>האם פירוש הדבר ש- RSA נפרצה?</u>

לא. ההתקן החדש מציע אפשרות לפרק לגורמים את המודולוס N יותר בזול ובפחות זמן מאשר על ידי שימוש ברשת של מחשבים העובדים במקביל, אבל הוא לא מאפשר פריצה של מפתחות RSA בגודל כלשהו. על מנת להתגונן בפני ההתקן, מספיק להשתמש במפתח גדול יותר. למעשה, שימוש במפתח בגודל של 768 סיביות כפי שמומלץ על ידי גורמים מוסמכים מבטיח חסינות בפני ההתקן החדש. הסיכון העיקרי שמעמיד ה-TWINKLE הוא האפשרות לפרוץ ביתר קלות למערכות אשר משתמשות בגודל מפתח של 512 סיביות לכל היותר.

10. פירוק לגורמים באמצעות מחשב קוונטי

מחשב קוונטי [18] הוא אמצעי אשר מסוגל לממש אלגוריתמים קוונטיים חדשניים אשר לא ניתן לממשם במחשבים ספרתיים (דיגיטליים) רגילים. לא מדובר בהאצת המהירות של מחשבים דיגיטליים קונבנציונליים אלא בהאצת המהירות האסימפטוטית של האלגוריתמים על ידי שימוש באפקטים קוונטיים טהורים.

במחשב קוונטי, האינפורמציה נטענת כמחרוזת של סיביות קוונטיות - "קיוביטים" (qubits). קיוביט הוא עצם קוונטי, למשל אטום (או יון) המסוגל לאכלס מצבים קוונטים שונים. שני מצבים כאלה יכולים לשמש לצורך אגירה של מידע ספרתי (דיגיטלי). למשל, אטום במצב היסוד שלו (ground state) מתאים לערך "0" של הקיוביט. אותו אטום במצב מעורר (excited state) מתאים לערך "1" של הקיוביט. עד עתה, אין שום דבר חדש בהשוואה למחשבים ספרתיים רגילים פרט לצפיפות של המידע הספרתי.

ואולם, היתרון העיקרי של מחשב קוונטי **לא** קשור לצפיפות הקיוביטים. בעוד שמחשוב קוונטי מתחיל ומסתיים עם מידע המקודד לביטים בינאריים, החישוב עצמו מבוצע במציאות המסתורית של הקוונטים, שבה מערכת פיזיקלית יכולה להיות במצב שידוע כחפיפה של מצבים (superposition of states). עבור אטום יחיד, ניתן להפיק מספר אינסופי של מצבים חופפים. יכולת זו של קיוביטים מאפשרת לייצג בו-זמנית את כל הערכים האפשריים של משתני הקלט. כאשר מחשב קוונטי מבצע צעד אחד, זה כאילו שהוא עשה זאת עבור כל ערך קלט אפשרי בבת אחת. לכן, מחשבים קוונטיים זקוקים לצעד אחד בלבד כדי לפתור בעיות בכל גודל שהוא. זהו המצב האולטימטיבי בעיבוד מקבילי: מקביליות ברמת הביט.

באמצעות שימוש בתהודה מגנטית גרעינית, ניתן למדוד האם החלקיקים המרכיבים את האטום – הפרוטונים והנויטרונים בגרעין שלו – מסתובבים בכיוון זה או אחר. מדידה כזו מהווה את שלב הקלט של החישוב, מאחר שברגע שסיבוב החלקיק נצפה, סיבוב של חלקיק אלמנטרי הופך לקבוע באחד משני המצבים הבינאריים שלו: סיבוב כלפי מעלה או סיבוב כלפי מטה.

הלוגיקה מתבצעת בכל מחשב קוונטים כאשר אטומי הקיוביט שלו משפיעים על הסיבוב של אטומי קיוביט שכנים. כאשר אטום של מחשב קוונטים בנוי במבנה מתאים, הוא יכול לבצע מספר פעולות מתמטיות במקביל.

אלגוריתם פרקטי המבוסס על מחשוב קוונטי לצורך פירוק לגורמים של מספרים גדולים, פורסם לראשונה ב-1994 על ידי פיטר שור מ-AT&T Research. מאחר שהצפנה ממלאת תפקיד מרכזי בביטחון לאומי, התוצאה של שור הובילה לסבב של השקעות ממשלתיות במחקר בנושא מחשבים קוונטים.

<u>פרוק לגורמים של מספרים שלמים באמצעות האלגוריתם של שור (Shor)</u>

אתאר את פעולת האלגוריתם על המספר $N=30$. ראשית, בוחרים באופן אקראי מספר y, כך שהגורם המשותף המקסימלי של המספרים y ו- N שווה ל-1 ($GCD(y,N)=1$). נתייחס לפונקציה המחזורית,

$$f(x)=y^x \pmod N \qquad x=0,\ 1,\ 2,\ 3,\ \ldots .$$

עבור $N=30$, למשל, נבחר באקראי את המספר $y=11$. נקבל,

$$f(0) = 1 \pmod{30} = 1,$$
$$f(1) = 11 \pmod{30} = 11,$$
$$f(2) = 11^2 \pmod{30} = 1.$$

המחזור T של הפונקציה $f(x)$ הוא בבירור, $T=2$. ניתן למצוא מחזור זה על ידי שימוש באלגוריתם הקוונטי לחישוב טרנספורם פורייה הדיסקרטי (ראה [18]). על מנת למצוא גורם של המספר N, מחשבים $z=y^{T/2}=11^1=11$. הגורם המשותף הגדול ביותר של $(z+1,N)=(12,30)$ הוא 6. הגורם המשותף הגדול ביותר של $(z-1,N)=(10,30)$ הוא 10. שני המספרים 10 ו-6 הם גורמים של 30. בדרך זו מוצאים שני גורמים של המספר N.

שיטת פירוק לגורמים זו נכשלת לפעמים. למשל, כאשר T הוא מספר אי זוגי. ניתן להראות, אם זאת, שאם y נבחר באקראי, ההסתברות לכישלון היא נמוכה. בפרט, במקרה שלנו, $N=30$, לפונקציה $f(x)=y^x \pmod{30}$, יש מחזור זוגי עבור כל y ($1<y<30$),

$$T = 2, \quad y = 11,\ 19,\ 29$$
$$T = 4 \quad y = 7,\ 13,\ 17,\ 23$$

בחישוב שלנו, יש למצוא את הגורם המשותף הגדול ביותר של שני המספרים, N ו- y. ניתן לעשות זאת בצורה יעילה באמצעות האלגוריתם של אוקלידס.

11. סיכום

במהלך 23 השנים מאז הצגת מערכת ההצפנה RSA פותחו מספר שיטות התקפה מעניינות מאוד, אולם עדיין לא נמצאה התקפה הרסנית ממש. ההתקפות שהתגלו עד היום, בעיקר מדגימות את הצורך בהכרת המלכודות מהן צריך להימנע כאשר משתמשים במערכת. כיום נראה, ששימוש נכון במערכת RSA מעניק הגנה טובה מאוד על המידע המוצפן.

חילקנו את ההתקפות הידועות על RSA למספר קטגוריות:

1) **התקפות אלמנטריות:** התקפות אלו מנצלות שימוש לקוי במערכת, כמו למשל שימוש במודולוס משותף או חתימה על הודעה בלתי מוכרת (סינוור).
2) **מפתח פרטי קטן או מפתח פומבי קטן:** על מנת להקטין את הזמן הדרוש לביצוע הצפנה, פענוח או חתימה, ובמיוחד במכשירים דלי הספק (כדוגמת כרטיסים חכמים), קיים פיתוי להשתמש בערכים קטנים של מפתח פרטי או מפתח ציבורי, דבר אשר פותח פתח להתקפות קשות על המערכת.
3) **התקפות מימוש:** התקפות כנגד מימוש מסוים של מערכת RSA כמו, למשל, מימוש המתבסס על משפט השאריות הסיני לשיפור הביצועים. מכאן ניתן ללמוד, שלא מספיק לחקור את המבנה המתמטי של המערכת אלא צריך להתייחס גם לצורת המימוש.
4) **פירוק לגורמים של המודולוס:** בעשור האחרון חלו תמורות גדולות בפיתוח שיטות לפירוק לגורמים של מספרים שלמים. שתי סיבות עיקריות לכך הן פיתוח של אלגוריתמים חדשניים לפירוק לגורמים (נפת שדה המספרים NFS והנפה הריבועית QS), והאפשרות להשתמש במחשוב מבוזר (distributed computing)[24].
5) **שימוש בחומרה מיוחדת (למשל, ה-TWINKLE):** ה-TWINKLE הנו התקן אופטי מהיר במיוחד והוא מאפשר לשפר את המהירות של האלגוריתמים QS ו-NFS לפירוק לגורמים בשלושה סדרי גודל (עבור אותה עלות וזמן).

ראינו שניתן למנוע חלק מההתקפות על ידי "ריפוד" (padding) אקראי מתאים של ההודעה המקורית לפני ביצוע ההצפנה או החתימה.

הסיכון העיקרי, לדעתי, העומד בפני RSA כיום הוא ההתפתחויות המתמידות והמהפכניות בטכניקות פירוק לגורמים של מספרים שלמים גדולים. מצד אחד, פיתוח אלגוריתמים מתוחכמים יותר, ומצד שני השימוש במחשוב מבוזר מסיבי ופתיחת האפשרות לשימוש במחשוב אופטי מהיר וזול. האפשרות האחרונה שקולה כנגד השימוש במספר גדול של מחשבים במקביל. מכאן הצורך להקפיד ולהשתמש במפתחות בגודל מתאים.

[24] למשל ברשת האינטרנט.

12. נספח א – תוכנית ליצירת מספרים ראשוניים

מובאת להלן תוכנית בשפת C++ אשר יוצרת מספרים ראשוניים עד למספר נתון.

```
/*********************************************************
*
* June 2000
* Implementing the Generation of Primes
*
*********************************************************/
#include <iostream.h>
#include <math.h>

#define MAX_PRIMES 10000000

int numPrimes;
int primes[MAX_PRIMES];

void generatePrimes(int num, int display)
{
     int isPrime,sq;
     numPrimes=0;
     for (int i=2;i<num;i++)
     {
          // test i for primeness
          isPrime=1;
          sq=sqrt(i);
          for (int k=0;k<numPrimes;k++)
          if (sq>=primes[k])
          {
               if (i%primes[k]==0) // not prime - has
                                   primes factor
               {
                    isPrime=0;
                    break;
               }
          }
          else
               break;
          if (isPrime)
          {
               primes[numPrimes++]=i;
               if (numPrimes==MAX_PRIMES) return;
               if (display) cout << i << "\t";
          }
     }
}
```

```
void main()
{
     int p;

     cout << "This is just to show the program works
:\n";
     generatePrimes(100,1);
     cout << "\n";

     generatePrimes(p=100000,0);
     cout << "This is the number of primes up to "<<p<< "
: "<<numPrimes;
     cout << "\n";

     generatePrimes(p=1000000,0);
     cout << "This is the number of primes up to "<<p<< "
: "<<numPrimes;
     cout << "\n";

     generatePrimes(p=10000000,0);
     cout << "This is the number of primes up to "<<p<< "
: "<<numPrimes;
     cout << "\n";
}
```

Output

This is just to show the program works:
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97

This is the number of primes up to 100000 : 9592
This is the number of primes up to 1000000 : 78498
This is the number of primes up to 10000000 : 664579

13. נספח ב – תוכנית לפירוק לגורמים ראשוניים

מובאת להלן תוכנית בשפת C++ אשר מבצעת פירוק לגורמים של מספרים גדולים בשיטת פולרד-רו (Pollard's Rho).

```
/*********************************************************
*
* July 2000
* Implementing Pollard's Rho Factoring Method
*
*********************************************************/
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int gcd(int u, int v)
// This function uses Euclids algorithm to compute a gcd
{
     if (v==0) return u;
          else return gcd(v,u-u/v*v);
}

int factoring(int n)
// This is the function which perform the factoring
{
     int i=1;
     int x=(rand()%n),y=x; // x and y are "random
numbers"
     int k=2;
     int d; // stores the gcd which could be a factor
     while (1)
     {
          i=i+1; // counter
          x=(x*x-1)%n; // picking another random x
          if (y>x)
               d=gcd(y-x,n);// checking for a common
factor
          else
               d=gcd(x-y,n);
          if (d!=1 && d!=n) return d; // found factor
          if (i==k)
               { y=x;k=2*k;}
     }
}

int getPrime()
```

```
// This function generates a prime number using a
relatively brute
// force method of testing randomly generated int's for
primeness.
{
     int temp;
     while (1)
     {
          temp= rand();
          int sqrtTemp=(int)sqrt(temp);
          for (int ch=2;ch<sqrtTemp+1;ch++) // one need
only test to halfway
          if (temp%ch==0) break; // has a factor
          if (ch==sqrtTemp+1) return temp;
     }
}

void main()
{
     int x,y;

     srand( (unsigned)time( NULL ) );

     for (int i=0;i<4;i++)
     {
          x=getPrime();
          y=getPrime();
          cout << "Primes are " << (double)x << "," <<
(double)y << endl;
          cout << "One of the factors is : " <<
factoring(x*y)<< "\n\n";
     }
}
```

Output

Primes are 30241,24169
One of the factors is : 30241
Primes are 9473,6947
One of the factors is : 6947
Primes are 24469,4919
One of the factors is : 4919
Primes are 25867,9377
One of the factors is : 9377

14. נספח ג – שברים משולבים

בנספח זה נציג סקירה של קירוב מספרים רציונליים על ידי שברים משולבים (continued fractions). לפרטים נוספים ראה [19].

שבר משולב מוגדר על ידי

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots \cfrac{1}{a_n}}},$$

ומסומן על ידי $[a_0, a_1, \ldots, a_{n-1}, a_n]$. המספרים $[a_0, a_1, \ldots, a_m]$, $0 \le m \le n$ נקראים **המתכנסים** (convergents) לשבר המשולב למעלה. אנו נתייחס לשברים משולבים פשוטים וחיוביים בלבד – כאלה אשר עבורם a_i הם שלמים וחיוביים.

משפט ג.1

השבר הנמשך $[a_0, a_1, \ldots, a_{n-1}, a_n] = [a_0, a_1, \ldots, a_{n-1} + \frac{1}{a_n}] = [a_0, \ldots, a_{m-1}, [a_m, \ldots, a_n]]$

עבור $0 \le m \le n$.

הוכחה: ההוכחה באינדוקציה מיידית. ♦

משפט ג.2

יהי $n > r \ge 1$, $p_0 = a_0,\ p_1 = a_0 a_1 + 1,\ p_{r+1} = a_{r+1} p_r + p_{r-1}$, ו- $q_0 = 1, q_1 = a_1, q_{r+1} = a_{r+1} q_r + q_{r-1}, n > r \ge 1$. המתכנס ה-r-י, $[a_0, a_1, \ldots, a_r]$,

של $[a_0, a_1, \ldots, a_{n-1}, a_n]$ נתון על ידי $\frac{p_r}{q_r}$.

הוכחה: ההוכחה באינדוקציה מיידית. ♦

המשפט האחרון מאפשר לחשב את המתכנסים בדרך פשוטה. ניתן להוכיח שהמתכנסים המתקבלים בצורה זאת הם שברים מצומצמים. כמוכן, ממשפט ג.1, נובע שניתן להניח ש- $a_n > 1$. שהרי, אם $a_n = 1$ אז $[a_0, a_1, \ldots, a_{n-1}, 1] = [a_0, \ldots, a_{n-1} + 1]$. במשפט הבא (המובא ללא הוכחה) נראה כי הייצוג הנ"ל הוא יחיד.

משפט ג.3

יהיו $x=[a_0,a_1,\ldots,a_n]=[b_0,b_1,\ldots,b_m]$ שני שברים משולבים פשוטים וחיוביים, המקיימים $a_n>1$ וגם $b_m>1$. אזי, $m=n$ וגם $a_i=b_i$, $0\le i\le n$. ♦

כעת נראה כי כל מספר רציונלי ניתן לייצוג על ידי שבר נמשך.

משפט ג.6

יהי $\frac{h}{k}$ שבר מצומצם. קיים שבר נמשך אשר ערכו שווה ל- $\frac{h}{k}$.

הוכחה. נשתמש באלגוריתם של אוקלידס לחישוב המחלק המשותף הגדול ביותר (GCD).

יהי $h=a_0k+r_0$.

אזי,

$$\frac{h}{k}=a_0+\frac{1}{\frac{k}{r_0}}.$$

אם $\frac{k}{r_0}$ הוא לא שלם, חזור על הפעולה עבור $\frac{k}{r_0}$, עצור כאשר אין שארית. התהליך זהה לאלגוריתם של אוקלידס למציאת GCD. ♦

בצורה זאת ניתן לחשב את המתכנסים. ההתקפה של ווינר (Weiner) מבוססת על כך ש- $\frac{k}{d}$ הוא אחד מהמתכנסים הללו. על מנת להוכיח זאת, טוענים כי המתכנסים מקרבים את השבר קרוב יותר מאשר כל שבר אחר בעל מכנה קטן או שווה לשבר הנ"ל (להוכחה, ראה [19]).

15. רשימת מקורות

[1] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, IT-22: 644-654, 1976.

[2] R.L.Rivest, A.Shamir and L.Adleman. *A method for obtaining digital signatures and public-key cryptosystems*, Communication of the ACM 21(2):120-126, Feb. 1978.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction To Algorithms*, McGraw-Hill & MIT Press, 2001.

[4] Seberry J and Pieprzyk J., *Cryptography: An introduction to Computer Security*, Prentice Hall, Australia, 1989.

[5] Stallings William, *Practical Cryptography for Data Internetworks*, Computer Society Press, California, 1996.

[6] Jan C.A, *Basic Methods of Cryptography*, Cambridge University Press, UK, 1998.

[7] D.Boneh and R.Venkatesan, *Breaking RSA may not be equivalent to factoring*, EUROCRYPT '98, Lecture Notes in Computer Science, vol. 1403, Springer-Verlag, Berlin and New-York, 1998, pp. 59-71.

[8] C.Pomerance, *A tale of two sieves*, Notices Amer. Math. Soc. 43 (1996), p. 1473-1485.

[9] Lenstra H., Verheul R., *Selecting Cryptographic Key Sizes*, November 1999.

[10] Boneh Dan, *Twenty Years of Attacks on the RSA Cryptosystem*, Notices of The AMS, February 1999, pp. 203-213.

[11] D.Boneh and G.Durfee, *New results on cryptanalysis of Low private exponent RSA*, preprint, 1998.

[12] Jean-Francois, *How (Not) to Design RSA Signature Schemes*, Lecture Notes in Computer Science, 1431, 1998, pp. 14-28.

[13] R.L.Rivest, A.Shamir and L.Adleman. *A method for obtaining digital signatures and public-key cryptosystems*, Communication of the ACM 21(2):120-126, Feb. 1978.

[14] *The Effectiveness of Lattice Attacks Against Low-Exponent RSA* Christophe Coup'e, Phong Nguyen, and Jacques Stern.

[15] P.Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, CRYPTO '96, Lecture Notes in Computer Science, vol.1355, Springer-Verlag, Berlin and New York, 1997, pp. 131-142.

[16] D.Boneh, R.DeMillo, and R.Lipton, *On the importance of checking cryptographic protocols for faults*, EUROCRYPT'97, Lecture Notes in Computer Science, vol. 1233, Springer-Verlag, Berlin and New-York, 1997, pp. 37-51.

[17] Adi Shamir, *Factoring Large Numbers with the TWINKLE Device*, Proceeding of EUROCRYPT, 1999.

[18] Gennady P.Berman et al, *Introduction to Quantum Computers*, World Scientific, 1998.

[19] G.H.Hardy & E.M. Wright, *An Introduction to the Theory of Numbers*, Oxford Clarendon Press, 1975.