האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

סמינר בקריפטוגרפיה
20374

TWINKLE

***T**he **W**eizmann **In**stitute **K**ey **L**ocating **E**ngine*

implementation of

Quadratic Sieve Factoring

העבודה הוכנה על-ידי: גדי וישנה 027388412
בהדרכתו של: ד"ר מיקולינסקי ולדימיר

תאריך ההגשה: *5* אוגוסט *2004*

תוכן

תוכן *2*

מבוא *3*

The Rivest-Shamir-Adelman Encipherment System *4*

מעט רקע מתמטי ל-*RSA* *4*

האלגוריתם - *RSA* *5*

הדגמה *7*

כיוון ההתקפה על *RSA* *7*

פירוק מספר גדול לגורמים *8*

חיפוש ישיר *8*

הטכניקה של פרמה *8*

שרשראות האופניים של להמר (*Lehmer [1926]*) *8*

הניפוי הריבועי - *The Qaudratic Sieve* *11*

העקרונות של *QS* *11*

הגדרת בסיס הגורמים וטווח הניפוי *11*

הניפוי -*Sieving* *12*

שלב המטריצה *13*

סיבוכיות *14*

*NFS*מול *QS* *15*

TWINKLE *16*

מוטיבציה *16*

תיאור כללי *16*

מבנה ה- *TWINKLE* *16*

ראש בראש - *TWINKLE*מול *PC* *18*

סיכום *18*

ביבליוגרפיה *19*

מבוא

פירוק מספר גדול לגורמים זו בעיה שמתמטיקאים רבים הקדישו לה זמן רב. בתחילה השיטה היחידה לפירוק מספר היתה לנסות לחלק בכל המספרים הראשוניים הקטנים מהשורש שלו. זו היתה הדרך היחידה עד שפרמה הציע טכניקה שמתבססת על המשוואה הריבועית: $a^2-b^2=(a+b)(a-b)$. השיטה של פרמה היתה למצוא את הריבוע הקטן ביותר שגדול מהמספר אותו רוצים לפרק כך שההפרש בין השניים יהיה גם הוא מספר ריבועי. ברגע שיש לנו ריבוע גדול מהמספר כך שההפרש גם הוא ריבוע, אפשר לחשב את המחלקים של המספר ע"י המשוואה הריבועית.

הטכניקה של פרמה אומנם מהווה שיפור משמעותי ביחס לשיטת הניסוי, אבל מול המספרים שמשמשים היום מערכות הצפנה הטכניקה של פרמה אינה שונה משיטת הניסוי. מאז הטכניקה של פרמה, פותחו כמה שיטות מורכבות יותר שמפרקות מספרים ביעילות הרבה יותר גדולה, אבל עדיין אינן מגיעות לביצועים שיכולים לעמוד בזמן אמת במשימת הפירוק. השיטות היעילות ביותר כיום הן ה-*Quadratic Sieve* וה-*Number Field Sieve*. שתי שיטות אלו מבוססות על טכניקה הדומה מאוד לזו של פרמה.

במאמר זה נציג את שיטת ההצפנה *RSA* שכדי לפרוץ אותה יש לפרק מספר גדול לגורמים. נציג את המימוש של להמר לשיטה של פרמה –מימוש ע"י מכונת חישוב שאינה מחשב קונוונציונלי. המאמר ידון בעיקר בשיטת הפירוק - *Quadratic Sieve* – ובמימוש של השיטה. בסוף החלק על ה-*QS* יוצג ההבדל האסימפטוטי בין *NFS* ל-*QS*.

The Rivest-Shamir-Adelman Encipherment System

הטכניקה שפותחה ע״י ריבסט-שמיר-אדלמן *(RSA)* היא טכניקה מתמטית המשמשת להצפנה עם מפתחות ציבוריים.

הצפנה עם מפתחות ציבוריים מסתמכת על יכולת של צד א׳ לפרסם מנגנון הצפנה כך שכל צד ב׳ יוכל להצפין הודעה בעזרת אותו מפתח, ורק המפרסם (צד א׳) יוכל לפענח את ההודעה.

מנגנון ההצפנה שצד א׳ מפרסם הוא המפתח הציבורי של א׳.

מנגנון ההצפנה שצד א׳ שומר בסוד הוא המפתח הפרטי של א׳.

מעט רקע מתמטי ל-RSA

משפט פרמה

עבור p ראשוני: $x^{p-1}=1 \pmod{p}$, לכל x זר ל-p.

הערות

אפשר להפוך את המשפט לקריטריון פשוט לבדיקת ראשוניות. אם השוויון אינו מתקיים עבור x זר ל-p, אז בהכרח p אינו ראשוני. אם השוויון מתקיים, אז p נקרא *pseudoprime* לפי בסיס x, אבל הוא עדיין לא חייב להיות ראשוני.

למשל עבור 341, $2^{10}=1024=341\cdot3+1$ ולכן $2^{10}=1 \pmod{341}$, וממילא $2^{340}=(2^{10})^{34}=1 \pmod{341}$.

מצד שני, $341=11\cdot31$. אפילו אם p הוא פסאודו ראשוני ביחס לכל בסיס, זה עדיין לא מחייב ש-p יהיה ראשוני. הדוגמא הקטנה ביותר לזה היא $p=561$ שאינו ראשוני כי $561=3\cdot11\cdot17$, אבל $x^{560}=1 \pmod{561}$ לכל x זר ל-561. מספרים כאלה נקראים מספרי *Charmichael*, והם נדירים מאד.

בדרך כלל, בדיקה עבור $x=2$, $x=3$ ו-$x=5$ אמורה להיות בדיקה מספיק טובה. מכל הנ״ל נובע שההיפוך של משפט פרמה מהווה שיטה הסתברותית לבדיקת ראשוניות (כלומר, יתכן שהשיטה תכריז על מספר פריק כאילו הוא ראשוני).

קיימת שיטה וודאית לבדיקת ראשוניות *(AKS Primality Test)* שסיבוכיותה $O(\ln^{12}p)$. סיבוכיות זו אינה גבוהה כמו חלוקה בכל המספרים הראשוניים הקטנים מהשורש שזה $O(p^{½})$, אבל עדיין גבוהה מדי למימושים מהירים.

פונקציית אויילר

$\varphi(n)$ – מספר המספרים החיוביים הזרים ל-n.

משפטים רלוונטים

משפט 1

עבור $p_1,p_2,...,p_k$ ראשוניים, והמכפלה $n=\Pi p_i$ מתקיים $\varphi(n)=\Pi(p_i-1)$

מקרה פרטי

עבור p ו-q ראשוניים, והמכפלה $n=pq$ מתקיים: $\varphi(n)=(p-1)(q-1)$

משפט אוילר (הרחבה של משפט פרמה)

לכל x זר ל-n מתקיים: $x^{\varphi(n)}=1 \pmod n$

מקרה פרטי

עבור p ו-q ראשוניים המכפלה $n=pq$ ו- x זר ל-n מתקיים: $x^{\varphi(n)}=1 \pmod n$

משפט 2

לכל e זר ל-$\varphi(n)$: $f(x)=x^e \pmod n$ חד-חד ערכית.

מקרה פרטי

עבור p ו-q ראשוניים והמכפלה $n=pq$ ו- e זר ל-$\varphi(n)$: $f(x)=x^e \pmod n$ חד-חד ערכית.

משפט 3

לכל e זר ל-$\varphi(n)$ קיים d כך ש- $ed=1 \pmod{\varphi(n)}$ ומכאן ש- $(x^e)^d=x^{ed}=x^{1+C(\varphi(n))}=x \pmod n$.

(זה נובע מאלגוריתם אוקלידס המוכלל, לחישוב מחלק משותף מקסימלי; ראה להלן).

מקרה פרטי

עבור p ו-q ראשוניים והמכפלה $n=pq$. לכל e זר ל-$\varphi(n)$ קיים d כך ש- $ed=1 \pmod{\varphi(n)}$ ומכאן ש- $(x^e)^d=x^{ed}=x^{1+C(\varphi(n))}=x \pmod n$.

האלגוריתם - RSA

בניית מפתח ציבורי

הגרלת שני מספרים ראשוניים גדולים p ו-q

כדי למצוא מספר ראשוני גדול – ראשית מגרילים מספר – ואז מוודאים שהוא ראשוני, את ווידוא הראשוניות מבצעים ע"י היפוך משפט פרמה.

גודלם של המספרים p ו-q קובע את קושי הפירוק של n לגורמים.

שלב זה הוא שלב קריטי בבניית המפתחות – הטעות הפשוטה ביותר היא שימוש בפונקציות האקראיות של המהדר בו כותבים את מערכת ההצפנה – מספר זה גם אם הוא מאותחל ע"י גורמים אקראיים כמו זמן ריצת המערכת או אחרים אינו מהווה מספר אקראי אמיתי וקל יחסית לשחזר את הראשוניים ה"רנדומליים" (במקום מספר ראשוני בן 500 ביטים יש לחפש רק את נקודת המוצא בת 32 הביטים).

מציאת e זר ל-$\varphi(n)$

זרות באה לידי ביטוי ע"י: $gcd(\varphi(n),e)=1$

ב-$\frac{2}{3}^2$ מהמקרים 3 יתאים, וב-$\frac{4}{5}^2$ 5 יתאים... כך שנוח לבחור e קטן מאד. אפשר גם לקבוע את e מראש,

ולאלץ את הראשוניים p,q להיות כאלה ש- $p-1,q-1$ זרים ל- e.

המפתח הציבורי הוא הצירוף של המכפלה $n=pq$ ו-e.

חישוב המפתח הפרטי

לפי משפט 3 קיים d כך ש- $e\cdot d=1 \pmod{\varphi(n)}$

נשתמש באלגוריתם אוקלידס המורחב למציאת d מתוך $\varphi(n)$ ו-e.

אלגוריתם אוקלידס המורחב

האלגוריתם $GCD(a,b)$ מוצא את המחלק המשותף הגדול ביותר של a ו-b.

האלגוריתם המורחב מוצא α ו-β שיקיימו: $\alpha\cdot a+\beta\cdot b=GCD(a,b)$.

$$
\begin{array}{lll}
\multicolumn{3}{l}{Extended\_GCD(a,b)} \\
 & if & b=1 \\
 & then & return\ (0,1) \\
 & else & (\varphi,\alpha) \leftarrow Extended\_GCD(b, a \bmod b) \\
 & & return\ \left(\alpha, \dfrac{1-\alpha\cdot a}{b}\right)
\end{array}
$$

בעזרת אלגוריתם אוקלידס המורחב ניתן בקלות למצוא d שיקיים את המשוואה:

$$\alpha\cdot\varphi(n)+d\cdot e=GCD(\varphi(n),e)=1$$

המפתח הפרטי הוא d.

צורת השימוש ב-*RSA*

השולח (הצד המצפין) משתמש במפתח הציבורי (N,e) של הנמען

המידע המקורי – x.

המידע המוצפן – $x^e \pmod n$

לפי משפט 2 המעבר הזה הוא חד-חד ערכי, ולכן מידע זה ניתן לפיענוח.

הנמען (הצד המפענח) משתמש במפתח הפרטי שלו (N,d)

המידע המוצפן שהתקבל הוא – $x^e \pmod n$.

המידע המפוענח הוא – $(x^e)^d \pmod n = x^{ed} \pmod n = x \pmod n$

הדגמה

נגריל שני מספרים ראשוניים: $p=631,\ q=419$.

נחשב את המכפלות: $n=631\cdot 419=264389,\ \varphi(n)=630\cdot 418=263340$

נחפש e זר ל-n - 13 הוא המספר הקטן ביותר שמתאים.

נמצא עכשיו d מתאים:

נפעיל את האלגוריתם *Extended_GCD* על $\varphi(n)$ ו- e ונקבל $(\alpha,\beta)=(-1,20257)$ ו- $d=\beta=20257$.

המפתח הציבורי הוא: $(n=264389,e=13)$

המפתח הפרטי הוא: $d=20257$.

כאשר X רוצה לשלוח הודעה עבור Y, הוא משתמש במפתח הציבורי.

X רוצה לשלוח 324, לשם כך הוא מחשב את $324^{13}\ (\mathrm{mod}\ n)$ ושולח את המספר 108100.

Y מקבל את ההודעה, מחשב $108100^{20257}(\mathrm{mod}\ n)$ ומקבל 324.

כיוון ההתקפה על **RSA**

המידע החסר לנו בבואנו לנסות ולפענח הודעה שהוצפנה ב-*RSA* הוא המפתח הפרטי d. הנתונים העומדים לרשותנו הם n ו-e מתוך המפתח הציבורי.

לכן כיוון ההתקפה שלנו יהיה: מציאת d מתוך n ו-e.

את d ניתן לחשב מתוך $\varphi(n)$ ו-e. למעשה, מכיוון ש- e ידוע, מתוך $\varphi(n)$ קל למצוא גם את d. *יתרה מזו,* $n-\varphi(n)=p+q-1$ *ולכן מידיעת* n *ו-* $\varphi(n)$ *אפשר למצוא את הגורמים* p *ו-* q *על-ידי פתרון משוואה ריבועית.*

לכן, הסיכוי היחיד למצוא את d *הוא אם נדע את* $\varphi(n)$.

בשביל $\varphi(n)$ צריך לדעת את p ו-q, אבל יש לנו רק את מכפלתם n.

ננסה אפוא למצוא את הגורמים הראשוניים של n (כלומר את p ו-q).

(נעיר כאן שלא הוכח עדיין שהתקפה מוצלחת על *RSA* שקולה ליכולת לפרק את n. מאידך, לא ידועה שום התקפה אחרת כאשר המסרים המוצפנים נבחרים באופן זהיר).

פירוק מספר גדול לגורמים

חיפוש ישיר

ננסה לחלק את n ב-$2, 3, 5, \ldots, \sqrt{n}$

הסיבוכיות ישרה ביחס לגודלו של n, עבור n בן x ספרות נסרוק כ- $10^{½x}$ מספרים בטרם נגיע לפיתרון, לכן זוהי פעולה בלתי ישימה עבור מספרים גדולים.

הטכניקה של פרמה

p ו-q הם ראשונים גדולים ולכן אי זוגיים $\Leftarrow$ $x=½(p+q)$ ו- $y=½(p-q)$ הם שלמים,

ומתקיים : $n=x^2-y^2$.

ננסה למצוא x ו-y כך ש- $n=x^2-y^2$ ואז $x-y$ ו-$x+y$ הם הגורמים של n.

x צריך להיות גדול מ-$\sqrt{n}$ וחייב להיות ריבועי, לכן נבחר x (ו-y בהתאם) כך :

$$x^2 = g(a) = \left(a + \left\lceil \sqrt{n} \right\rceil\right)^2$$

$$y^2 = f(a) = \left(a + \left\lceil \sqrt{n} \right\rceil\right)^2 - n$$

ננסה $a=0,1,2,\ldots$ עד שנקבל $f(a)$ ריבועי (כלומר y שלם).

שרשראות האופניים של להמר (Lehmer [1926])

המכונה של להמר נבנתה ב-1926 והיתה מסוגלת לבדוק 60 מספרים בשניה.

מכונה זו מימשה את הטכניקה של פרמה.

אנחנו צריכים למצוא – $f(a)$ שניתן להוציא ממנו שורש, ואז לחשב ע"י השורשים של $f(a)$ ו-$g(a)$ את המחלקים של n.

אורכה של כל שרשרת הוא מספר ראשוני (קטן משורש-n).

כל חוליה בשרשראות מסמנת את הסימן של Legendre[1] עבור $f(a)$ מעל הראשוני המתאים.

הסימן של Legendre הוא מחזורי ובערך במחצית המצבים 1.

כאשר $f(a)$ בכל השרשראות מקבל 1 – כלומר $f(a)$ במודולו p (אורך השרשרת) הוא שארית של מספר ריבועי – אנו מנסים לחשב את השורש של $f(a)$, ומקווים להצליח.

למה זה עובד?

לכל p ראשוני ניתן לכתוב כל x ריבועי כך: $x=(\alpha p+\varepsilon)^2 \quad (\varepsilon<p)$

ואז: $x=\varepsilon^2+2\varepsilon\alpha p+a^2p^2\equiv\varepsilon^2 (\mathrm{mod}\ p)$

כלומר ל-x יש שארית ריבועית לכל אורך שרשרת שנבחר.

לכן כש-$f(a)$ הוא ריבועי, בכל השרשראות יהיה 1.

בכל השרשאות יהיה 1 רק באחד מתוך 2^n (n הוא מספר השרשראות) מקרים, לכן מספר השרשראות יכול להקטין את מספר הבדיקות של מספרים שאינם באמת ריבועיים.

הדגמה של פעולת השרשראות של Lehmer

עבור: $n=264389$

ושרשראות באורכים הבאים: 3, 5, 7, 11, 13.

סימן Legendre יופיע במחזוריות על השרשראות:

$=a$	0	1	2	3	4	5	6	7	8	9	10	11	12
3	0	1	0	0	1	0	0	1	0	0	1	0	0
5	1	0	0	0	0	1	0	0	0	0	1	0	0
7	0	0	1	1	1	0	0	0	0	1	1	1	0
11	0	0	0	0	0	1	1	0	0	1	1	0	0
13	1	0	1	1	0	0	0	1	1	0	1	0	0

---

[1] Legendre symbol

$$\left(\frac{a}{p}\right)=(a\,|\,p)\equiv\begin{cases}0 & \text{If } p|a.\\ 1 & \text{if } a \text{ is a quadratic residue modulo } p.\\ -1 & \text{if } a \text{ is a quadratic nonresidue modulo } p.\end{cases}$$

כלומר אם קיים x עבורו $x^2=a\ (\mathrm{mod}\ p)$, נסמן: $\left(\frac{a}{p}\right)=0$

(עבור p ראשוני, לעולם לא נקבל $\left(\frac{f(a)}{p}\right)=0$.)

(מתוך: http://mathworld.wolfram.com/LegendreSymbol.html)

נחשב את f ו-g עבור ערכי a שונים ונבדוק את ההתאמה לפי הסימן של *Legendre* המופיע על השרשראות:

			הסימן על השרשראות				
a	$f(a)$	$g(a)$	3	5	7	11	13
0	836	265225		1			1
1	1867	266256	1				
2	2900	267289			1		1
3	3935	268324			1		1
4	4972	269361	1		1		
5	6011	270400		1		1	
6	7052	271441				1	
7	8095	272484	1				1
8	9140	273529					1
9	10187	274576			1	1	
10	11236	275625	1	1	1	1	1

עבור $a=10$ קיבלנו $f(a)$ שהוא שארית ריבועית בכל השרשראות.

ובחישוב השורש אכן נקבל ש- $\sqrt{f(a)}=\sqrt{11236}=106$

נמשיך לפי הטכניקה של פרמה:

$$y=\sqrt{f(a)}=106$$
$$x=\sqrt{g(a)}=\sqrt{275625}=525$$
$$(x+y)(x-y)=631\cdot 419=264389$$

ומצאנו את הגורמים של n.

סיבוכיות

השיטה של פרמה אינה משפרת את הסיבוכיות האסימפטוטית, והמימוש של להמר אומנם משיג תוצאות יפות אבל אסימפטוטית עדיין המשימה קשה מדי.

הניפוי הריבועי - The Qaudratic Sieve

שיטת הניפוי *Qaudratic Sieve* היא פיתוח של קארל פומרנץ *(Carl Pomerance)* משנת 1981, הוא למעשה הרחיב רעיונות קודמים של קרייטצניק *(Kraitchnik)* ודיקסון *(Dixon)*.
ה-*QS* היתה השיטה המהירה ביותר לפירוק מספר לגורמים עד ל-*NFS* *(Number Field Sieve)* שפותחה ב- 1993. בגלל מורכבות ה-*NFS*, מספרים עם פחות מ-*100* ספרות עדיין עדיף לפרק בעזרת ה-*QS*.

העקרונות של QS

אם n הוא המספר אותו רוצים לפרק, השיטה *QS* היא לנסות למצוא x ו-y כך ש- $x^2 \equiv y^2 \pmod n$ ואז $(x+y)(x-y) \equiv 0 \pmod n$ ומכאן קל לחשב את $GCD(x-y,n)$ ולמצוא את אחד המחלקים של n.
(בהסתברות של ½ מתקיים $x \not\equiv \pm y \pmod n$ לכן יתכן שתוצאת ה-GCD תהיה n או *1*, ואז צריך יהיה למצוא עוד זוג מספרים כאלו.)

לשם כך נגדיר: $Q(x) = \left(x + \lfloor \sqrt{n} \rfloor\right)^2 - n = \tilde{x}^2 - n$

ונחשב את $Q(x_1), Q(x_2), \ldots, Q(x_k)$. בסעיפים הבאים יוסבר כיצד למצוא x_i מתאימים.

מתוך קבוצת ה-$Q(x)$ אנחנו מחפשים תת-קבוצה כך ש: $Q(x_1)\cdot Q(x_2)\cdot\ldots\cdot Q(x_r)=y^2$

כיוון ש- $Q(x)\equiv\tilde{x}^2 \pmod n$

אז גם $Q(x_1)\cdot Q(x_2)\cdot\ldots\cdot Q(x_r)\equiv(\tilde{x}_1\cdot\tilde{x}_2\cdot\tilde{x}_3\cdot\ldots\cdot\tilde{x}_r)^2 \pmod n$

ומצאנו x ו-y מתאימים לדרישות.

הגדרת בסיס הגורמים וטווח הניפוי

אנחנו צריכים דרך יעילה למצוא x_i כך שהמכפלה $\prod Q(x_i)$ תתן מספר ריבועי. כדי שזה יתקיים צריך שכל גורם ראשוני המחלק את המכפלה יחלק אותה מספר זוגי של פעמים. לשם כך צריך לפרק לגורמים ראשוניים כל אחד מ-$Q(x_i)$ המרכיבים את המכפלה.
כדי לפשט את הבדיקה אנחנו מעוניינים ש-$Q(x_i)$ יהיו קטנים ככל האפשר, ויתחלקו בראשוניים מתוך קבוצת ראשוניים ידועה לנו – לקבוצה זו נקרא **בסיס הגורמים** *(Factor Base)* ונסמן אותה ב-*B*. גודלה של הקבוצה *B* משפיע על ביצועי האלגוריתם ולכל וריאנט של ה-*QS* יש לחשב את ה-*B* האופטימלי לפי גודלו של n.

נסמן לשימוש בהמשך – $B = \{p_1=2;\ p_i \text{ is the smallest prime bigger than } p_{i-1} \mid p_1, p_2, \ldots, p_k\}$

ומובן שגודל הקבוצה B הוא k.

כדי ש-$Q(x)$ יהיה קטן אנחנו צריכים לבחור x קרוב ל-0. אם ננסה לקחת x קרוב ל-0 מלמטה נקבל מספרים הקרובים מאוד ל-n לכן $x>0$. – נגדיר איפה את **טווח הניפוי** ע"י $[1,M]$.

שיפורים אפשריים

אם נוסיף לבסיס הגורמים גם את -1, אז מספרים הקרובים ל-n מלמטה גם הם מספרים קטנים יחסית, וניתן לבחור x שלילי ועדיין לקבל מספרים שקל יחסית לעבוד איתם. כלומר טווח הניפוי הוא: $[-M,M]$.

את בסיס הגורמים אפשר להקטין ע"י בדיקת היתכנות –

אם $p|Q(x)$ אז: $\left(x+\lfloor\sqrt{n}\rfloor\right)^2=Q(x)+n\equiv n\pmod{p}$

לכן צריך להכניס לבסיס הגורמים רק מספרים ראשוניים המקיימים עבור x מתוך טווח הניפוי את התנאי: $\left(x+\lfloor\sqrt{n}\rfloor\right)^2\equiv n\pmod{p}$. זוהי בדיקה בסיבוכיות נמוכה לפי משפט *ההיפוך הריבועי* של גאוס[2], שמאפשר לחשב את סימן לגרנז' בזמן לוגריתמי (באלגוריתם דומה לזה של אוקלידס).

הניפוי - Sieving

הגדרה: המספר X חלק *(Smooth)* מעל הקבוצה Y אם לכל p גורם ראשוני של X מתקיים ש- p איבר של Y.

ההגדרות: "X חלק מעל B" ו-"X הוא p_k חלק" – שקולות (p_k הוא האיבר הגדול בקבוצה B).

בשלב הניפוי אנחנו מחפשים $Q(x)$ שיכולים להיות חלק מקבוצה המרכיבה מכפלה שהיא מספר ריבועי. לשם כך אנחנו צריכים לעבור על x מתוך טווח הניפוי, לחשב את $Q(x)$ ולבדוק האם $Q(x)$ חלק מעל B. עבור כל אחד מהמספרים שמתקבלים, אנו צריכים לבדוק אם הוא חלק ע"י מעבר וחלוקה בבסיס הגורמים. פעולה סדרתית כזו אינה מעשית מבחינת זמן ביצוע. לכן נעבוד על בסיס הגורמים במקביל.

לכל p בבסיס הגורמים אם $p|Q(x)$ אז גם $p|Q(x+p)$.

ובכיוון ההפוך, אם $x\equiv y \pmod{p}$ אז גם $Q(x)\equiv Q(y) \pmod{n}$.

לכל p נפתור: $Q(x)=s^2\equiv 0\pmod{p}, x\in Z_p$

---

[2] *Quadratic Reciprocity Theorem of Gauss* - משפט ההיפוך הריבועי של גאוס

If p *and* q *are distinct odd primes, then the congruences $x^2\equiv q \pmod{p}$ and $x^2\equiv p \pmod{q}$ are both solvable or both unsolvable unless both* p *and* q *leave the remainder 3 when divided by 4 (in which case one of the congruences is solvable and the other is not).*

Written symbolically: $\left(\frac{p}{q}\right)\cdot\left(\frac{q}{p}\right)=(-1)^{\frac{1}{4}\cdot(p-1)\cdot(q-1)}$

Where: $\left(\frac{p}{q}\right)\equiv\begin{cases}1 & \text{for } x^2\equiv p \pmod{q} \text{ solvable for } x\\ -1 & \text{for } x^2\equiv p \pmod{q} \text{ not solvable for } x\end{cases}$ *is known as a* ***Legendre symbol***

(מתוך: http://mathworld.wolfram.com/QuadraticReciprocityTheorem.html)

את המשוואה הזאת ניתן לפתור ע״י האלגוריתם של Shanks-Tonelli[3]. פתרון המשוואה הריבועית יתן לנו שני שורשים, נסמן אותם ב- s_{1p} ו- $s_{2p}=p-s_{1p}$. מכאן אנו רואים ש-$Q(x_i)$ עבור x_i מתוך טווח החיפוש מתחלק ב-p כאשר $x_i=s_{1p},s_{2p}+pk$ עבור k שלם.

כעת בתהליך שמזכיר מאד את מכונת השרשראות של להמר, אנחנו מתקדמים ולכל x אנחנו בודקים באלו ראשוניים מבסיס הגורמים $Q(x)$ מתחלק. כדי לבצע את החיפוש הזה על כל טווח הניפוי כדאי לחלק את העבודה למספר מחשבים במקביל – וכל מחשב יקבל חלק מטווח הניפוי.

אנחנו בודקים האם $Q(x)$ הוא חלק, אם לא – זורקים אותו ועוברים ל-x הבא, ואם כן שומרים אותו בתוך מטריצה שתוגדר בהמשך.

שיפורים אפשריים

פעולת חילוק היא פעולה מורכבת. לכן במקום לעשות חישוב מדוייק, כדי לבדוק במהירות את ה"חלקות" של $Q(x)$ נבצע הערכה באמצעות מספר הביטים של הגורמים ש-$Q(x)$ מתחלק בהם, וכך במהירות גבוהה נקבל תוצאות וודאיות במרבית המקרים, ובאלו שיש לגביהם ספק – פשוט זורקים ועוברים הלאה, לא חסרים לנו מספרים לבדוק.

שלב המטריצה

בשלב זה אנחנו מחזיקים קבוצה Q של מספרים גדולים $Q(x_i)$ – ואנו צריכים למצוא קבוצה חלקית ל-Q כך שמכפלת האיברים של הקבוצה החלקית תתן מספר ריבועי. לשם כך נגדיר וקטור של חזקות שמייצג מספרים חלקים מעל בסיס הראשוניים שלנו B.

עבור m מספר חלק מעל B: $m=\prod_{i-1}^{k} p_i^{v_i} \Rightarrow v(m)=(v_1,v_2,\dots,v_k)$.

אנו צריכים למצוא קבוצת וקטורים המייצגים מספרים מתוך Q כך שמכפלת המספרים תהיה מספר ריבועי. לשם כך אנחנו צריכים למצוא קבוצת וקטורי חזקות שסכומם יתן וקטור שכל איבריו זוגיים.

---

[3] *The Shanks-Tonelli algorithm is a procedure for solving a congruence of the form* $x^2 \equiv n \pmod p$*, where* p *is an odd prime and* n *is a quadratic residue of* p.
In other words, it can be used to compute modular square roots.
First find positive integers Q *and* S *such that* $p-1 = 2^S \cdot Q$. *where* Q *is odd. Then find a quadratic nonresidue* W *of p and compute* $V \equiv W^Q \pmod p$*. Then find an integer* n' *that is the multiplicative inverse of* n *(mod p) (i.e.,* $n \cdot n' \equiv 1 \pmod p$*).*
Compute $R \equiv n^{\frac{1}{2}(Q+1)} \pmod p$ *and find the smallest integer* $i \geq 0$ *that satisfies* $(R^2 \cdot n')^{2^i} \equiv 1 \pmod p$*.*
If i=0*, then* x=R*, and the algorithm stops.*
Otherwise, compute: $R' \equiv R \cdot V^{2^{(S-i-1)}} \pmod p$ *and repeat the procedure for* R=R'.
(מתוך: http://planetmath.org/encyclopedia/ShanksTonelliAlgorithm.html)

נבנה מטריצה V של וקטורי חזקות, המייצגים $Q(x)$ חלקים מעל B. ועכשיו אנו צריכים למצוא וקטור בינארי e שהכפלתו במטריצה תתן וקטור שכל איבריו זוגיים.

ע"י החילוץ הגאוסיאני (*gaussian elimination*) ניתן לפתור את המטריצה, ולקבל וקטור e מתאים.

שיפורים אפשריים

כדי לעשות זאת קל יותר לחישוב נמיר את המטריצה V למטריצה בינארית V_2. ואז וקטור $e\neq 0$ שמקיים $e\cdot V_2=0$ יהווה פתרון גם עבור המטריצה V.

חשוב לציין שוקטורים מודולו 2 אינם ייצוג חח"ע של וקטורי חזקות – אולם החסכון בזמן חישוב ובגודל הזכרון – אפילו שהוא לינארי בלבד – משתלם למרות הצורך בהחזקת מיפוי בין הוקטור הבינארי למספר המקורי שהוא מייצג.

מספר דוגמאות לייצוג וקטורי מעל הבסיס $\{2,3,5,7,11,13,17\}$:

$$3,651,921=2^0\cdot 3^2\cdot 5^0\cdot 7^4\cdot 11^0\cdot 13^2\cdot 17^0=v(0,2,0,4,0,2,0)\equiv v(0,0,0,0,0,0,0) \pmod 2$$
$$11,662=2^1\cdot 3^0\cdot 5^0\cdot 7^3\cdot 11^0\cdot 13^0\cdot 17^1=v(1,0,0,3,0,0,1)\equiv v(1,0,0,1,0,0,1) \pmod 2$$
$$1,071=2^0\cdot 3^2\cdot 5^0\cdot 7^1\cdot 11^0\cdot 13^0\cdot 17^1=v(0,2,0,1,0,0,1)\equiv v(0,0,0,1,0,0,1) \pmod 2$$

סיבוכיות

בסיס ראשוניים הכולל את כל הראשוניים עד n יביא לכך שכל $Q(x)$ הוא חלק מעל B, וכך תהיה הסיבוכיות למציאת $Q(x)$ מתאים $O(1)$ בלבד. אולם, גודלה של המטריצה (כ- $n/log(n)$) מונע מאיתנו להנות מהיעילות של השלב הראשון. מצד שני, בסיס ראשוניים קטן מאוד (למשל הראשוניים עד 1000) ייצר לנו מטריצה קטנה ונוחה לפתרון, ועל כך נשלם בסיכוי זעיר למצוא $Q(x)$ חלק מעל B.

נסמן ב- $\varphi(X,B)$ את מספר המספרים החלקים מעל B בטווח $[1,X]$. הסיכוי שמספר אקראי בתחום $[1,X]$ יהיה חלק מעל B הוא $\varphi(X,B)/X$, ולכן כדי למצוא מספר אחד מתאים אנחנו צריכים לעבור על $X/\varphi(X,B)$ מספרים אקראיים. חשוב להדגיש שהמספרים $Q(x)$ אינם אקראיים במובן הפורמלי, אבל כל הניתוחים ההיוריסטיים של שיטות הפירוק נעשים בהנחה שהם אקראיים במידה מספקת. כיוון שאנחנו צריכים $|B|=k$ מספרים כאלו לבניית המטריצה (כדי להבטיח פתרון לא טריוויאלי), אנחנו צריכים לעבור על $k\cdot X/\varphi(X,B)$ מספרים. עלות הבדיקה שמועמד $Q(x)$ הוא חלק מעל B היא ליניארית ב- B, ולכן העבודה הכוללת בייצור היא $k^2\cdot X/\varphi(X,B)$ פעולות (של נסיון חילוק מספר בגודל $O(n^{½})$ בראשוני קטן).

כפי שמובא במאמר של פומרנץ, כדי לקבל את המינימום עבור הנוסחא הזו צריך שהאיבר הגדול ב-B, אותו סימנו ב- p_k, יהיה קרוב ל- $e^{\frac{1}{2}\sqrt{\log(X)\cdot\log(\log(X))}}$, והעבודה הנדרשת היא: $e^{2\sqrt{\log(X)\cdot\log(\log(X))}}$, כלומר כ- $p_k^{\,4}$.

אבל מהו X?

באלגוריתם QS אנחנו מייצרים $Q(x)$ מסדר גודל של $n^{½+\varepsilon}$ כאשר ε קטן מאד (מכיוון שמספר המועמדים שנבדוק הוא חזקה קטנה של n). לכן, סדר הגודל של שלב הניפוי הוא: $e^{\sqrt{2\cdot\log(n)\cdot\log(\log(n))}}$.

לאחר שהנתונים נאספו במטריצה (בגודל k על k), צריך למצוא פתרון לא טריוויאלי, בעלות של $O(k^3)$ (האלימינציה של גאוס). למעשה החלק הזה של האלגוריתם הוא מהיר בהרבה, מכיוון שהמטריצה דלילה. בכל אופן מדובר בסיבוכיות זניחה ביחס לשלב איסוף המשוואות.

NFS* מול *QS

שיטה נוספת לפירוק מספר לגורמים היא ה-*Number Field Sieving*, זוהי שיטה מהירה יותר אסימפטוטית מה-*Quadratic Sieve*, אבל מסובכת הרבה יותר. שתי השיטות מחולקות לשני שלבים שלב הניפוי ושלב מטריצה.

סדר הגודל של שלב הניפוי ב-*NFS* הוא: $e^{c\cdot\log^{1/3} n\cdot\log^{2/3}\log n}$, כאשר c משתנה לפי סוגי הניפוי המדוייק בו משתמשים (ברוב המימושים: $2 \geq c \geq 1½$).

אסימפטוטית זהו סדר גודל קטן יותר משל *QS*, אבל במספרים "קטנים" (עד 100 ספרות) עדיף להשתמש ב-*QS* בגלל המסובכות של *NFS*.

TWINKLE

מוטיבציה

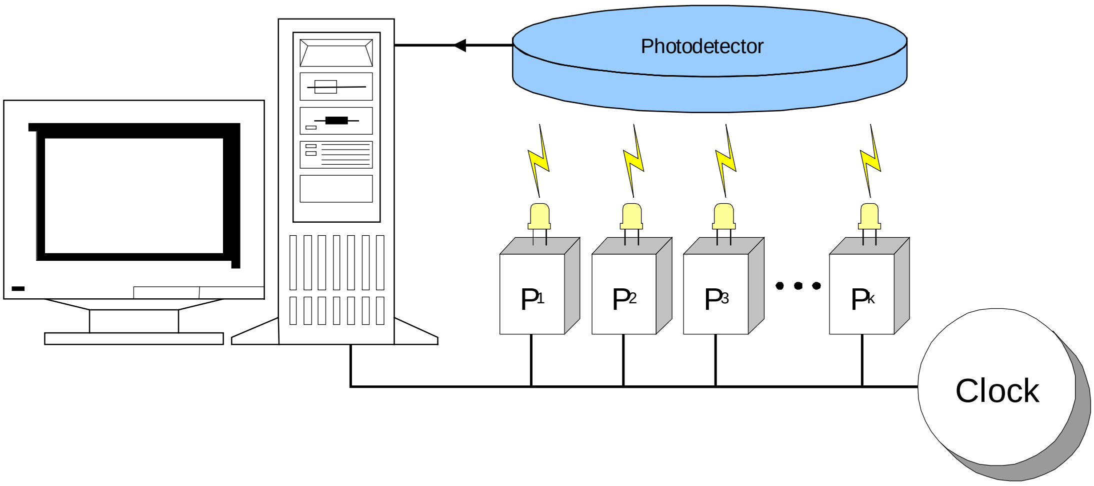
השלב הראשון, שלב הניפוי של ה-*QS* צורך איסוף משוואות רבות עבור השלב השני, שלב המטריצה. אולם, איסוף המשוואות איטי מאד ולו בגלל שצריך המון מועמדים לכל משוואה. חיפוש המועמדים מוגבל על-ידי מהירות השעון במחשבים שלנו, לפיכך כשמנסים לפרק בשיטה זו מחלקים את הניפוי בין מחשבים רבים.

לעומת זאת פרופ' עדי שמיר, מציע להחליף את המחשבים (האיטיים והיקרים) בלדים (זולים ומהירים). במקום לחלק את טווח הניפוי בין מחשבים שונים נחלק את בסיס הראשוניים בין לדים. הלדים זולים מאוד ומהירות מאוד.

תיאור כללי

ה-*TWINKLE* הוא מכשיר אלקטרואופטי, המורכב בתוך גליל שחור אטום בקוטר של כ-15 סנטימטרים ובגובה 25 ס"מ. תחתית הצינור מלאה בדיודות מאירות *(LEDs)* שמהבהבות בקצבים שונים, ובראש הגליל נמצא מד-אור *(photodetector)* שמודד את עוצמת האור של כל הדיודות יחד. מד האור מחובר למחשב ומודיע כשעוצמת האור מתאימה לעוצמה מסויימת שחושבה מראש. האיתות הזה הוא הפלט היחיד של ה-*TWINKLE*. האיתותים שה-*TWINKLE* מעביר מעידים על חלקות *(smoothness)* של מספר. כיוון שאירועים אלו מאוד נדירים המחשב יכול לאמת את חלקות המספר, ע"י חלוקה סדרתית בבסיס הגורמים, ולהמשיך את האלגוריתם *QS* במטרה למצוא את המחלקים של *n*.

מבנה ה- TWINKLE



TWINKLE scheme

כל דיודה משוייכת לראשוני מתוך בסיס הראשוניים B. השעון מודיע מהו x שאותו אנחנו בודקים כעת. דיודה i מאירה במחזור שעון x אם $p_i|Q(x)$. היחידה הצמודה לדיודה מכילה תא זכרון עם מונה מתאים כך שהדיודה מהבהבת בקצב מחזורי קבוע המתאים להתחלקות של $Q(x)$ ב-p_i.

עוצמת הדיודה הדרושה היא $log(p_i)$ כך שכאשר מאירות דיודות i ו-j אנו מקבלים עוצמת אור השווה ל-$log(p_i)+log(p_j)=log(p_i \cdot p_j)$. מכאן, כשעוצמת האור היא $log(Q(x))$ אנחנו יודעים ש-$Q(x)$ חלק מעל B. חשוב לציין (כיוון שאחת הבעיות המרכזיות במערכות של מספרים גדולים זו רמת הדיוק) שהיחס בין עוצמת האור של הדיודה החלשה ביותר לדיודה החזקה ביותר הוא סביר[4] וניתן לקבל אותו ע"י הוספת נגדים להקטנת המתח, ע"י בחירת דיודות בעוצמות שונות,ע"י הגדלת מספר הדיודות עבור המספרים הגדולים או ע"י הוספת משטח כהה שיסנן חלק מהאור עבור הראשוניים הקטנים.

בהדלקת המערכת יש לאתחל את הבקרים של הדיודות בפרמטרים של אורך המחזור ושל ההיסט בתחילת עבודת המכשיר (עבור $x=0$). כמו כן את החיישן האופטי צריך לכוון לעוצמת האור שתתאים ל-$log(Q(x))$. כיוון ש- $Q(x)$ מחושב לטווח חיפוש מסויים, צריך לחשב את הטווח המתאים ולא ערך יחיד. החיישן צריך להודיע על עוצמת אור הגדולה מ-$log(min(Q(x)))$. אם נחליף את הגבול התחתון של טווח הדיווח ב: $log(min(Q(x)))-log(p_k)=log(min(Q(x))/p_k)$, נקבל התראות גם עבור חלק מהמקרים בהם כמה ראשוניים מתוך הבסיס מחלקים את $Q(x)$ יותר מפעם אחת. הזזה נוספת של הגבול התחתון כלפי מטה תאפשר קבלת התראות על יותר מקרים חלקים מעל B אולם נקבל גם אירועי שגיאה שבהם מספרים שאינם בבסיס (ואין דיודה שמייצגת אותם) הם חלק מהגורמים.

אין צורך לחסום את עוצמת האור מלמעלה, כיוון שלא יתכן שסך הלוגריתמים של הגורמים של $Q(x)$ יהיה גדול יותר מהלוגריתם של $Q(x)$ עצמו.

פעולת הניפוי

השעון מעביר פולסים בקצב גבוה ($10GHz$), הנוריות מהבהבות בהתאמה לחלוקת $Q(x)$ בראשוניים המתאימים, והחיישן האופטי מעביר דיווח על זמן שעון בו עוצמת האור היתה מעל לקו התחתון שהוגדר. ה-*TWINKLE* מבצע את כל פעולת הניפוי במקביליות מרשימה. במקום לחלק את הראשוניים מהבסיס בין מחשבים שונים אנו באבחת שעון אחת סורקים את כולם. אולם חשוב לשים לב שמדובר על עבודה מהירה בלי שיפור אמיתי באלגוריתם, אסימפטוטית הסיבוכיות נשארה זהה.

כשמחשב מקבל את האיתות מה-*TWINKLE* הוא מחשב את $Q(x)$ המתאים (לפי מחזור השעון) ובודק מיהם הגורמים הראשוניים שלו. את השתתפות הגורמים האלו בפירוק $Q(x)$ הוא מסמן בוקטור בינארי כפי שהוסבר לעיל, וממשיך לשלב המטריצה.

---

[4] לפי חישובים מהסעיף הקודם $log(p_k)=½ \cdot (log(n) \cdot log(½ \cdot log(n)))^{½} \cdot log(e)$ שזה עבור n בן 500 ספרות קרוב ל-24.

ראש בראש - TWINKLE מול PC

עבור פירוק מספר מכל גודל שהוא, בניפוי ע"י *PC* מחלקים את טווח החיפוש לקבוצות בנות כ-*100,000,000* איברים כל אחת והמחשב סורק את כל הקבוצה בנסיון למצוא מספרים חלקים.

הזכרון הנדרש מהמחשב לצורך זה הוא של $n\cdot 100M$ ביטים, שזה – $0.1GB\cdot n/8$. אם נקבל כהנחה מוגזמת שזמן סריקת הזכרון מספיק כדי לדעת מי מהמספרים המאוחסנים שם חלק מעל *B*, נניח גם שאין צורך לגשת לדיסק הקשיח (האיטי יחסית), ובהתחשב בכך שמהירות הגישה לזכרון היא כ-*400MHz*, רק סריקת כל המספרים שבקבוצה אחת בפירוק של *512* ביט תיקח כ-*16* שניות.

לעומת זאת פעולת הניפוי ע"י ה-*TWINKLE* מתבצעת לפי מהירות שעון של *10GHz* וללא תלות בגודלם של המספרים. לפיכך את אותו קטע ניפוי שראינו שהמחשב מבצע ב-*16* שניות ה-*TWINKLE* יבצע ב-*0.01* שניות.

כלומר ה-*TWINKLE* מהיר פי *1600* מה-*PC*. (תוך התחשבות בהקלות בלתי מבוטלות שנעשו ל-*PC*)

סיכום

פירוק מספר לגורמים זוהי בעיה שעדיין אין לה פתרון סביר. אולם כבר מוכרים כמה אלגוריתמים תת-פולינומיאלים, שמאפשרים לנו להגיע לביצועים פירוק מהירים יחסית.

עדיין בקרב בין המצפינים לפורצים, ידם של המצפינים על העליונה. אולם *RSA-154 (512bits)* פחות בטוח ממה שחשבנו. ראוי להזכיר שהשיא של פירוק *RSA* הוא של *RSA-160 (530bits)*בשנת *2003*, הפירוק נעשה ע"י חלוקת הניפוי בין מחשבי *PC* רבים מכל רחבי העולם.

אחד הדברים היפים שניתן להסיק מהמאמר הזה זו היכולת של טכנולוגיה שאינה מחשב קונבנציונלי לבצע פעולות שנראות בלתי אפשריות במהירות שמחשב לא מסוגל להתקרב אליה. מכאן אנו יכולים גם להסיק שאין להמעיט בערכן של מכונות ייחודיות גם כאשר ברשותנו מחשבים חזקים מאד.

ביבליוגרפיה

[1] Konheim, A. G., "Criptograpy: A Primer", 1981, Chapter 8, PP. 294-330.

[2] Landquist, E., http://www.math.uiuc.edu/~landquis/
"MATH 488: Cryptographic Algorithms",
"The Quadratic Sieve Factoring Algorithm", December 14, 2001

[3] Pomerance, C., http://www.ams.com/notices/ (American Matematical Society)
"A Tale of Two Sieves", December, 1996.

[4] Shamir, A., http://www.windowsecurity.com/
"Factoring Largne Numbers with the TWINKLE device (Extended Abstract)", April 04, 2000

[5] http://mathworld.wolfram.com/

[6] http://planetmath.org/