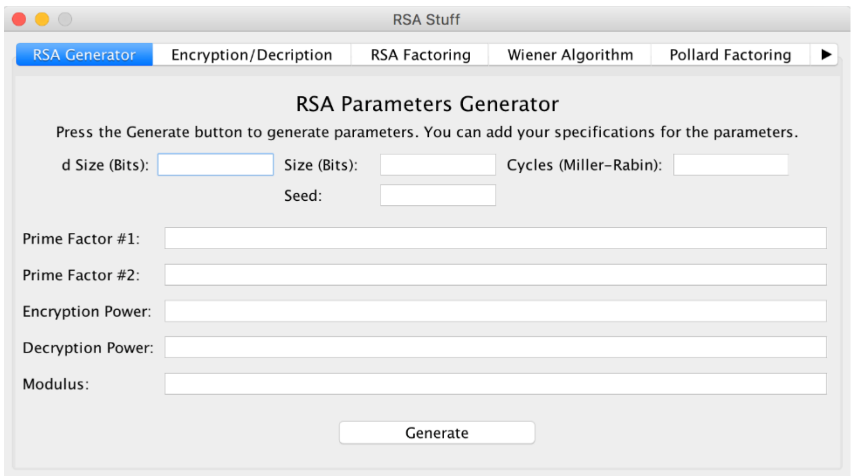
מדריך משתמש

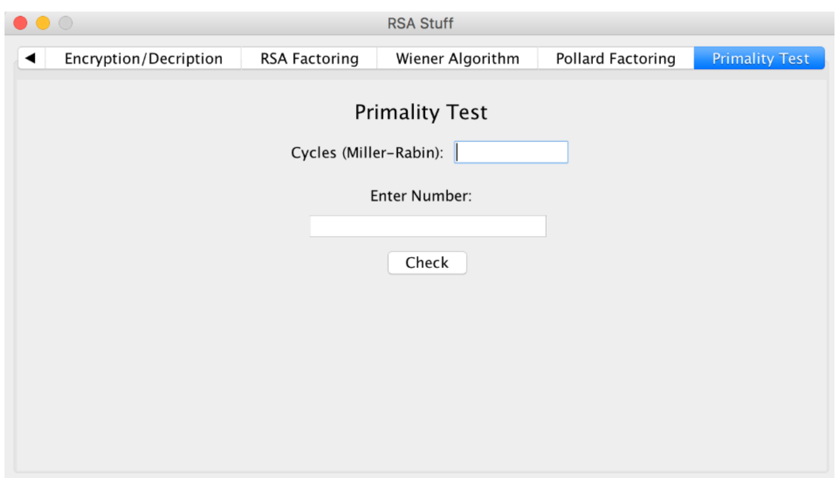
פונקציונאליות התכנה

יצירת הפרמטרים של RSA

התכנה מאפשרת לצור את הפרמטרים הנדרשים להצפנת RSA. התכנה משתמשת באלגוריתם מילר-רבין למציאת מספרים ראשוניים, ומאפשרת למשתמש לקבוע כמה "סיבובים" האלגוריתם יבצע (ככל שמתבצעים יותר סיבובים, קטן הסיכוי לכך שהמספר שיוגרל יהיה לא פריק). המשתמש יכול לקבוע כמה ביטים ידרשו לייצוג הבינארי של הגורמים של המודולוס. המשתמש יוכל להגדיר כמה ביטים ידרשו למפתח הפענוח (כדי להתאים את הפרמטרים לאלגוריתם וינר). עבור ערכים לא מתאימים (למשל, מספרים שליליים, אותיות) תוצג הערת שגיאה למשתמש, והערכים שילקחו בחשבון באלגוריתם יהיו ערכי ברירת המחדל – מספר הסיבובים באלגוריתם מילר-רבין הוא 1000, הגודל בביטים הוא 64 והגודל של חזקת הפענוח הוא 64.[1]

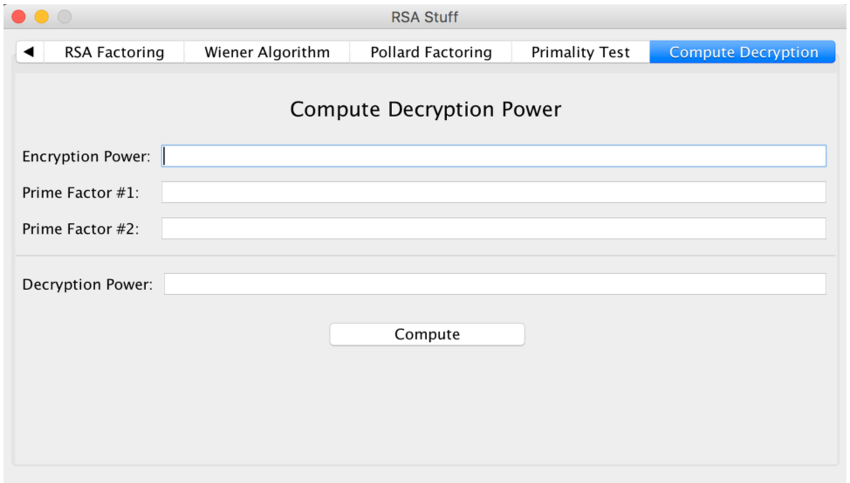


איור 1 מחולל פרמטרים



איור 2 בדיקת ראשוניות

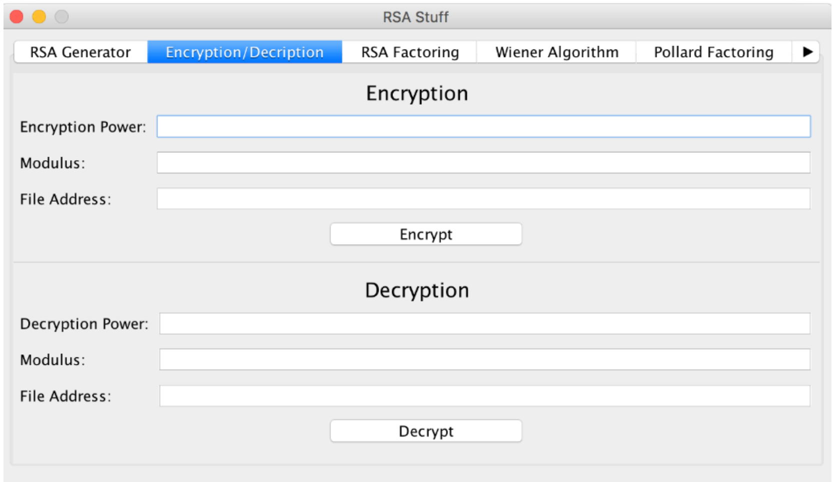
[1] הסרתי את השימוש ב-seed לקביעת ערכי ה-Random שמופיע בהצעת ההגשה, כדי שיבדקו ערכים רנדומלים שונים בכל פעם, ומחולל הפרמטרים לא יתקע על אותו מספר רנדומלי בכל פעם.



איור 3 חישוב חזקת הפענוח

הצפנה/פענוח של קבצים

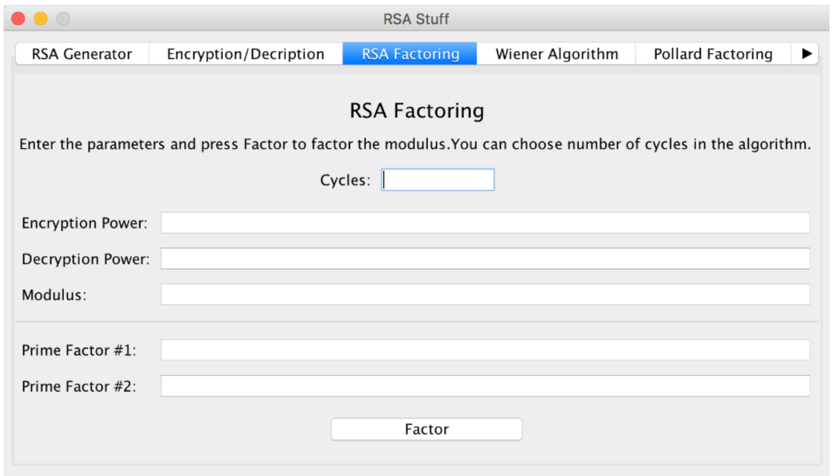
התכנה מאפשרת להצפין ולפענח קבצים בהינתן הפרמטרים להצפנה – חזקת ההצפנה/פענוח והמודולוס (חשוב לציין שבפועל לרוב לא משתמשים ב-RSA כדי להצפין קבצים, אלא מפתחות להצפנה סימטרית, מכיוון שפעולת ההעלאה בחזקה "יקרה" יותר מהפעולות הדרושות בהצפנה סימטרית מבחינת כוח החישוב הדרוש). התכנה מחלקת את הקובץ לבלוקים בגודל חצי ממספר הביטים הדרושים לייצוג המודולוס, ומצפינה כל בלוק לבלוק חדש בגודל מספר הביטים שדרוש ליצוג המדולוס. עבור ערכים לא מתאימים (למשל, ערכי חזקות הצפנה או מודולוס קטנים מדי) תוצג הודעת שגיאה למשמש. הקובץ שיתקבל כפלט מהפעולה ימצא באותה התיקיה כמו הקובץ שעליו הופעלה הפעולה. הרחבה על ההצפנה והפענוח מופיעה בחלק של דוגמת ההרצה.



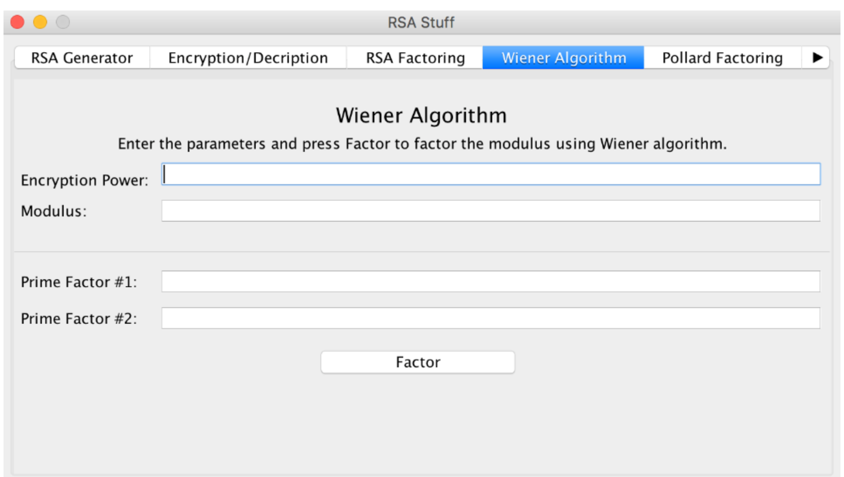
איור 4 הצפנה ופענוח בתכנה

פירוק המודולוס לגורמים

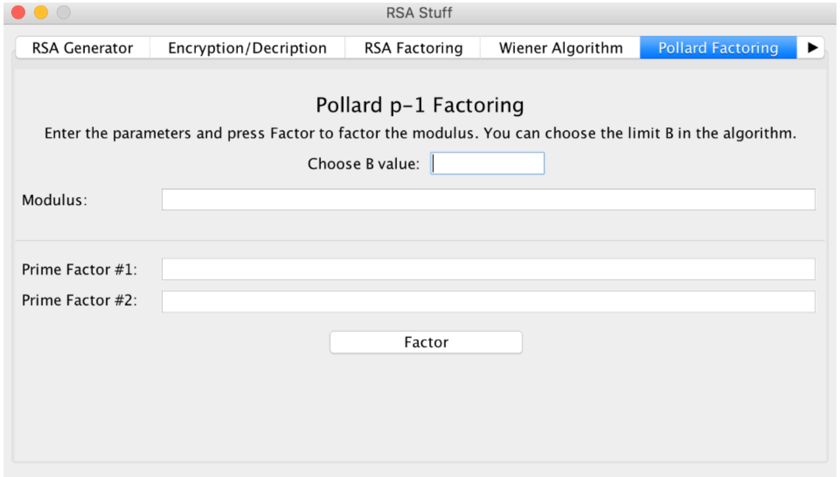
כדי לתקוף את RSA נרצה למצוא את חזקת הפענוח, בהינתן המפתחות הפומביים חזקת ההצפנה והמודולוס. דרך אפשרית לעשות זאת היא על ידי מציאת הפירוק של המודולוס לגורמים, ועל ידי כך לחשב את $\phi(n) = (p-1)(q-1)$. ניתן להגיע לחזקת ההצפנה על ידי $de = 1 \, mod\big(\phi(n)\big)$. התכנה מאפשרת להשתמש באלגוריתם $p-1$ של פולארד ואלגוריתם וינר המאפשרים לפרק לגורמים תחת הנחות מסוימות. בנוסף, מוצג אלגוריתם לפירוק המודולוס לגורמים בהינתן ערכי חזקות ההצפנה והפענוח.



איור 5 אלגוריתם לפירוק לגורמים בהינתן חזקת ההצפנה, הפענוח והמודולוס. כברירת מחדל, מספר הסיבובים הוא 1000.



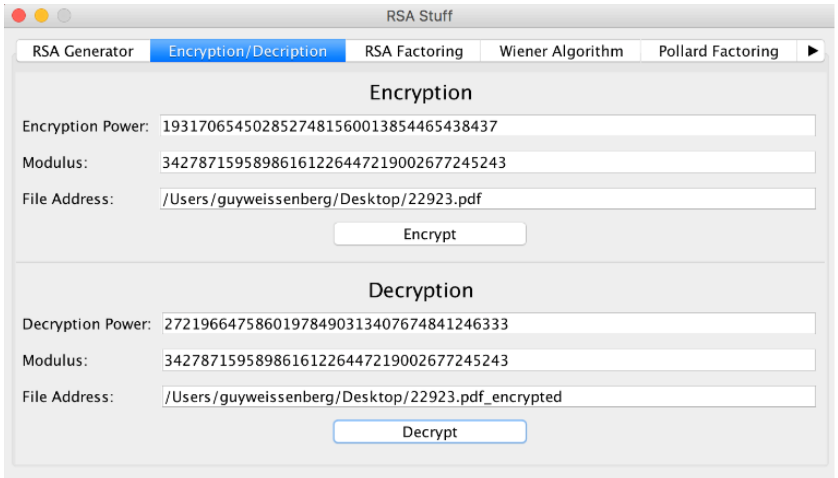
איור 6 אלגוריתם וינר לפירוק לגורמים



איור 7 אלגוריתם פולארד לפירוק לגורמים. כברירת מחדל החסם B הוא 10^6

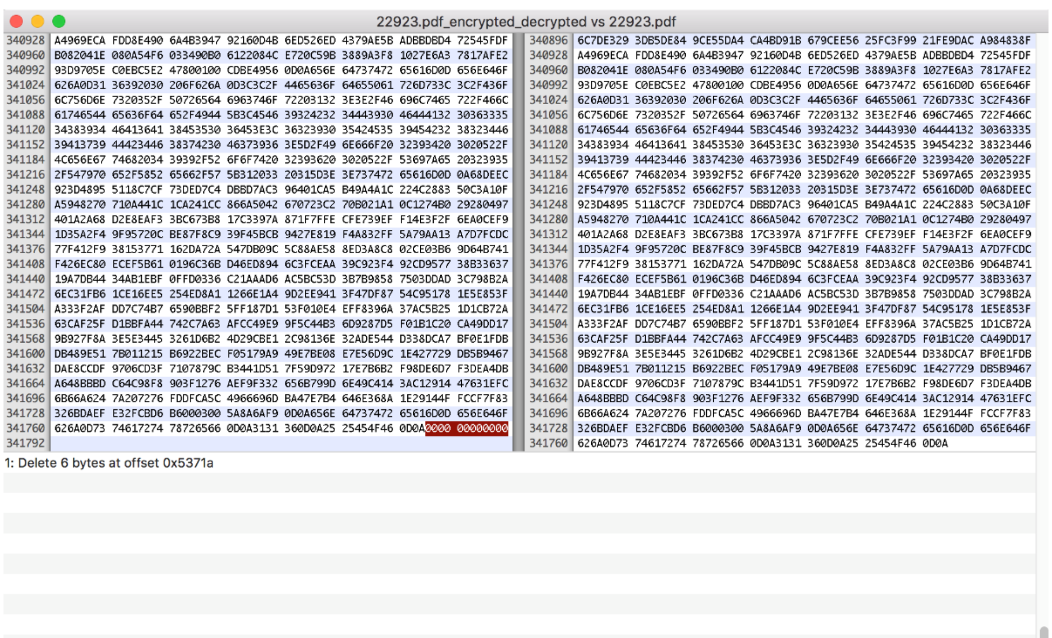
דוגמת הרצה

מצורפת דוגמת הרצה של הצפנת חוברת הקורס עם הפרמטרים הבאים :



הקובץ המוצפן זהה לשם הקובץ המקורי בתוספת _encrypted. בדוגמה, ההצפנה של הקובץ 22923.pdf תהיה הקובץ 22923.pdf_encrypted. הקובץ המפוענח יקבל את שם הקובץ המוצפן בתוספת _decrypted. בדוגמה, הפענוח של הקובץ 22923.pdf_encrypted יהיה הקובץ 22923.pdf_encrypted_decrypted.

השוואה בין התוכן הבינארי של הקובץ המופענח לקובץ המקורי מראה שהם זהים (למעט השלמת אפסים בסוף הקובץ כדי שגודל הקובץ יתחלק ב-8 בתים) :



הקובץ המקורי, הקובץ המוצפן והקובץ המפוענח נמצאים בתיקייה "דוגמת הרצה".

רקע תיאורטי

אלגוריתם מילר רבין

אלגוריתם מילר-רבין הוא אלגוריתם מונטה-קרלו[2] שבודק בזמן $O((\log n)^3)$ האם מספר הוא פריק. הסיכוי לשגיאה באלגוריתם הוא לכל היותר 0.25.

Algorithm 5.7: MILLER-RABIN(n)

```
write n − 1 = 2^k m, where m is odd
choose a random integer a, 1 ≤ a ≤ n − 1
b ← a^m mod n
if b ≡ 1 (mod n)
  then return ("n is prime")
for i ← 0 to k − 1
     ⎧ if b ≡ −1 (mod n)
  do ⎨   then return ("n is prime")
     ⎩   else b ← b^2 mod n
return ("n is composite")
```

איור 8 אלגוריתם מילר-רבין

נוכיח שהאלגוריתם מחזיר ש-n פריק רק כאשר הוא באמת פריק. נניח בשלילה שהאלגוריתם מחזיר n פריק עבור מספר ראשוני כלשהו. אם האלגוריתם אומר ש-n פריק, אז מתקיים $a^m \not\equiv 1\ mod\ n$. נסתכל על b באלגוריתם – מעלים את b בריבוע בכל חזרה של הלולאה. מכיוון שהאלגוריתם החזיר ש-n פריק, אז $a^{2^i m} \not\equiv -1\ mod\ n$ לכל $0 \le i \le k-1$. לפי ההנחה ש-n ראשוני, ממשפט פרמה נובע ש- $a^{2^k m} \equiv 1\ mod\ n$ (כיוון ש- $2^k m = n-1$). כלומר, $a^{2^{k-1}m}$ הוא שורש ריבועי של 1 מודולו n, ולכן הוא שווה ל-1 או -1. אך ערכו לא יכול להיות -1 שכן אז היה מוחזר ש-n פריק. לכן $a^{2^{k-1}m} \equiv 1\ mod\ n$. לכן גם $a^{2^{k-2}m}$ הוא שורש ריבועי של 1 מודולו n, ומאותם השיקולים מתקיים ש-$a^{2^{k-2}m} \equiv 1\ mod\ n$. נמשיך כך ונקבל ש-$a^m \equiv 1\ mod\ n$, אך זוהי סתירה שכן אז היה מוחזר ש-n ראשוני.

לכן אם האלגוריתם מחזיר ש-n פריק, הוא בוודאות פריק (אם האלגוריתם מחזיר ש-n ראשוני, הוא ראשוני בהסתברות מסויימת).

אלגוריתם $p-1$ של פולארד

אלגוריתם זה מצליח לחשב את הפירוק לגורמים של המודולוס בהינתן המודולוס וחסם B בהנחה שכל המחלקים הראשוניים של $p-1$ (כאשר p הוא אחד הגורמים הראשוניים) קטנים מהחסם B. בחירה של $B = 10^6$ תמצא כרבע מכלל המחלקים בעלי 12 ספרות.

[2] אלגוריתמים מסוג זה עשויים להחזיר תשובה לא נכונה לבעיה, בהסתברות מסוימת. במקרה של אלגוריתם מילר-רבין, האלגוריתם עשוי להחזיר שמספר ראשוני על אף שהוא לא כזה. הרצה חוזרת של האלגוריתם מקטינה את ההסברות לשגיאה.

```
Algorithm 5.8: POLLARD p − 1 FACTORING ALGORITHM(n, B)

a ← 2
for j ← 2 to B
  do a ← a^j mod n
d ← gcd(a − 1, n)
if 1 < d < n
  then return (d)
  else return ("failure")
```

איור 9 אלגוריתם p-1 של פולארד

נציג את הרעיון המרכזי באלגוריתם – נניח ש-p הוא אחד הגורמים הראשוניים של n, ונניח ש-$q \leq B$ לכל גורם ראשוני $q|(p-1)$. מכאן ש- $(p-1)|B!$. לאחר שנעלה את a בחזקה כך ש- $a \equiv 2^{B!} \ mod\ n$, נשים לב שמכיוון ש- $p|n$ חייב להתקיים $a \equiv 2^{B!} \ mod\ p$. מכאן ש- $2^{p-1} \equiv 1 \ mod\ p$. לפי משפט פרמה, $2^{p-1} \equiv 1 \ mod\ p$. מכיון ש- $(p-1)|B!$ הרי שמתקיים $a \equiv 1 \ mod\ p$, ולכן $p|(a-1)$. מכיוון ש- $p|n$ הרי ש- $p|d$ עבור $d = \gcd(a-1, m)$. כלומר, d הוא מחלק לא טריוויאלי של n (אלא אם $a = 1$). כיוון של-n יש שני מחלקים לא טריוויאלים, נוכל למצוא את השני בקלות.

פירוק לגורמים של המודולוס בהינתן חזקות ההצפנה והפענוח

אלגוריתם זה הוא אלגוריתם לאס-וגאס[3] המפרק את n לגורמים בהינתן חזקות ההצפנה והפענוח, בזמן פולינומיאלי. אלגוריתם זה מראה שאם התגלתה חזקת הפענוח עבור הצפנה בחזקה פומבית כלשהי, השימוש במודולוס איננו בטוח (גם עבור שימוש במפתח פומבי אחר), מכיוון שניתן לפרק את n לגורמים ולמצוא את חזקת הפענוח עבור כל מפתח פומבי.

```
Algorithm 5.10: RSA-FACTOR(n, a, b)

comment: we are assuming that ab ≡ 1 (mod φ(n))
write ab − 1 = 2^s r, r odd
choose w at random such that 1 ≤ w ≤ n − 1
x ← gcd(w, n)
if 1 < x < n
  then return (x)
comment: x is a factor of n
v ← w^r mod n
if v ≡ 1 (mod n)
  then return ("failure")
while v ≢ 1 (mod n)
  do { v0 ← v
     { v ← v^2 mod n
if v0 ≡ −1 (mod n)
  then return ("failure")
  else { x ← gcd(v0 + 1, n)
       { return (x)
comment: x is a factor of n
```

איור 10 פירוק לגורמים בהינתן חזקת הפענוח והמפתח הפומבי

---

[3] אלגוריתמים מסוג זה מספק תשובה נכונה בוודאות, אך הוא מצליח בהסתברות מסויימת. במקרים רבים, כמו גם באלגוריתם זה, האקראיות נועדה לשפר את זמן הריצה.

האלגוריתם מבוסס על מציאת השורשים הריבועיים של 1 מודולו n. אם x שורש לא טריוויאלי של 1 מודולו n מתקיים $x^2 \equiv 1^2 \; mod \; n$ וגם $x \not\equiv \pm 1 \; mod \; n$. בהינתן השורש, נוכל למצוא את המחלקים הלא טריוויאלים של n על ידי חישוב $\gcd(x+1,n)$ ו- $\gcd(x-1,n)$. כלומר, האלגוריתם מוצא שורשים לא טריוויאלים של 1 מודולו n, ומחשב את הפירוק של n לפיהם.

אלגוריתם וינר

אלגוריתם וינר מצליח לפרק לגורמים את המודולוס בהינתן המודולוס וחזקת הפענוח, כאשר מתקיים:

$$3d < n^{\frac{1}{4}}, \qquad q < p < 2q$$

כלומר, אם בייצוג הבינארי של n יש l ביטים, האלגוריתם יעבוד אם לחזקת הפענוח יש פחות מ-$\frac{l}{4} - 1$ ביטים בייצוג הבינארי, ו-p, q לא רחוקים מדי זה מזה.

נשים לב שלעיתים נרצה להשתמש בחזקת פענוח קטנה כדי להאיץ את תהליך הפענוח. בשימוש באלגוריתם יעילים להעלאה בחזקה, הפרמטרים שמקיימים את התנאים שלעיל יפחיתו את זמן הריצה של האלגוריתם ב-75%.

Algorithm 5.11: WIENER'S ALGORITHM(n, b)

$(q_1, \ldots, q_m; r_m) \leftarrow$ EUCLIDEAN ALGORITHM(b, n)
$c_0 \leftarrow 1$
$c_1 \leftarrow q_1$
$d_0 \leftarrow 0$
$d_1 \leftarrow 1$
for $j \leftarrow 2$ **to** m
do
  $c_j \leftarrow q_j c_{j-1} + c_{j-2}$
  $d_j \leftarrow q_j d_{j-1} + d_{j-2}$
  $n' \leftarrow (d_j b - 1)/c_j$
  comment: $n' = \phi(n)$ if c_j/d_j is the correct convergent
  if n' is an integer
  then
    let p and q be the roots of the equation
      $x^2 - (n - n' + 1)x + n = 0$
    if p and q are positive integers less than n
      then return (p, q)
return ("failure")

איור 11 אלגוריתם וינר

נציג את העקרונות הכלליים של האלגוריתם – יהיו $n = pq$ כאשר p, q ראשוניים $\phi(n) = (p-1)(q-1)$. מכיוון ש- $de = 1 \; mod \; (\phi(n))$, שמתקיים $de - t\phi(n) = 1$. בנוסף, מכיוון ש-$n = pq > q^2$ מתקיים $\sqrt{n} > q$.

מכאן ש- $0 < n - \phi(n) = q + p - 1 < 2q + q - 1 < 3q < 3\sqrt{n}$.

לכן – $\left|\frac{e}{n} - \frac{t}{d}\right| = \left|\frac{ed - tn}{an}\right| = \left|\frac{1 + t(\phi(n) - n)}{dn}\right| < \frac{3t\sqrt{n}}{dn} = \frac{3t}{a\sqrt{n}}$.

מכיוון ש-$t < d$ נקבל ש-$3t < 3d < n^{\frac{1}{4}}$. בנוסף, מכיוון ש-$3d < n^{\frac{1}{4}}$ נקבל: $\left|\frac{e}{n} - \frac{t}{d}\right| < \frac{1}{3d^2}$.

את $\frac{e}{n}$ נוכל לייצג כשבר משולב על ידי הסדרה $[q_1 \dots q_m]$ שנוכל לקבל מאלגוריתם אוקלידס (ערכים אלו הם המקדמים בכל שלב חלוקה באלגוריתם), באופן הבא : $q_1 + \frac{1}{q_2 + \cdots}$, כאשר הרכיבים הסופיים של השבר הם $q_1, q_1 + \frac{1}{q_2}, q_1 + \frac{1}{q_2 + \frac{1}{q_3}}$ וכו'.

ע"פ משפט שהוכיח ויינר, אם $\gcd(e,n) = \gcd(t,d) = 1$ ומתקיים $\left|\frac{e}{n} - \frac{t}{d}\right| < \frac{1}{2d^2}$, אז $\frac{t}{d}$ הוא אחד מהרכיבים הסופיים של השבר המשולב $\frac{e}{n}$. בהתבסס על משפט זה, כדי למצוא את $\frac{t}{d}$ עלינו לחפש את הרכיב הסופי המתאים של השבר המשולב $\frac{e}{n}$ (נעבור על כל הרכיבים הסופיים עד שנמצא את הרכיב המתאים). אם $\frac{t}{d}$ הוא רכיב סופי של $\frac{e}{n}$ נוכל לחשב את הערך של $\phi(n)$ על ידי $\phi(n) = \frac{ed-1}{t}$. לאחר ש-$\phi(n)$ נמצא נוכל את אחד הרכיבים הראשוניים של n על פי המשוואה $p^2 - (n - \phi(n) + 1)p + n = 0$ (המתקבלת משילוב ההגדרות של n ו-$\phi(n)$).

ביבליוגרפיה

D.R. Stinson, *Cryptography: Theory and Practice*, 3rd ed. (Chapman & Hall/CRC, 2006)

Wikipedia. *Wiener's attack*.
http://en.wikipedia.org/w/index.php?title=Wiener's%20attack&oldid=787941334 , 2017.

Wikipedia. *Pollard's p − 1 algorithm*.
http://en.wikipedia.org/w/index.php?title=Pollard's%20p%20%E2%88%92%201%20algorithm&oldid=781542828 , 2017.