

## מדריך משתמש

2	פונקציונאליות התכנה
2	יצירת הפרמטרים של RSA
3	הצפנה/פענוח של קבצים
4	פירוק המודולוס לגורמים
6	דוגמת הרצה
7	רקע תיאורטי
7	אלגוריתם מילר רבין
7	אלגוריתם $1 - p$ של פולארד
8	פירוק לגורמים של המודולוס בהינתן חזקות ההצפנה והפענוח
9	אלגוריתם וינר
11	בבליוגרפיה

## 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 שיוגדל 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 <sup>1</sup>64.

RSA Stuff

RSA Generator Encryption/Decription RSA Factoring Wiener Algorithm Pollard Factoring

**RSA Parameters Generator**

Press the Generate button to generate parameters. You can add your specifications for the parameters.

d Size (Bits):  Size (Bits):  Cycles (Miller-Rabin):

Seed:

Prime Factor #1:

Prime Factor #2:

Encryption Power:

Decryption Power:

Modulus:

Generate

איור 1 מחולל פרמטרים

RSA Stuff

Encryption/Decription RSA Factoring Wiener Algorithm Pollard Factoring Primality Test

**Primality Test**

Cycles (Miller-Rabin):

Enter Number:

Check

איור 2 בדיקת ראשוניות

<sup>1</sup> הסרתי את השימוש ב-seed לקביעת ערכי ה-Random שמופיע בהצעת ההגשה, כדי שיבדקו ערכים רנדומליים שונים בכל פעם, ומחולל הפרמטרים לא יתקע על אותו מספר רנדומלי בכל פעם.

RSA Stuff

◀ RSA Factoring Wiener Algorithm Pollard Factoring Primality Test Compute Decryption

Compute Decryption Power

Encryption Power:

Prime Factor #1:

Prime Factor #2:

Decryption Power:

Compute

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 בלוק חדש בגודל מספר הביטים שדרוש לייצוג המודולוס. 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.

RSA Stuff

RSA Generator Encryption/Decryption RSA Factoring Wiener Algorithm Pollard Factoring ▶

Encryption

Encryption Power:

Modulus:

File Address:

Encrypt

Decryption

Decryption Power:

Modulus:

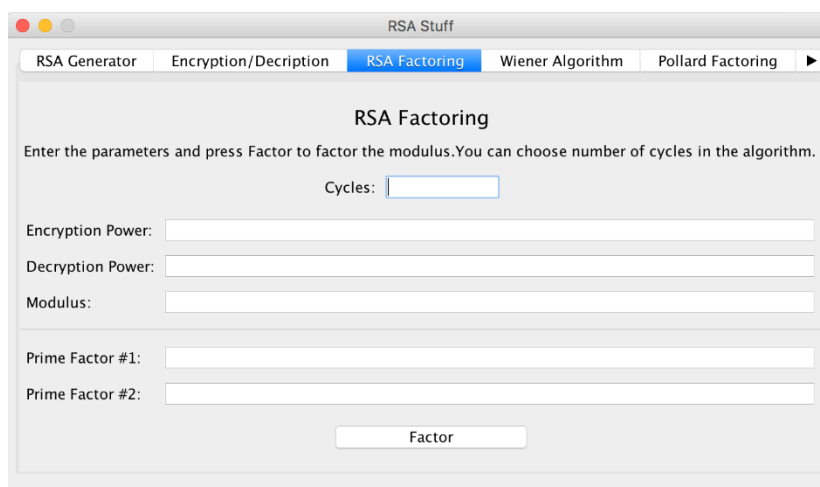
File Address:

Decrypt

איור 4 הצפנה ופענוח בתכנה

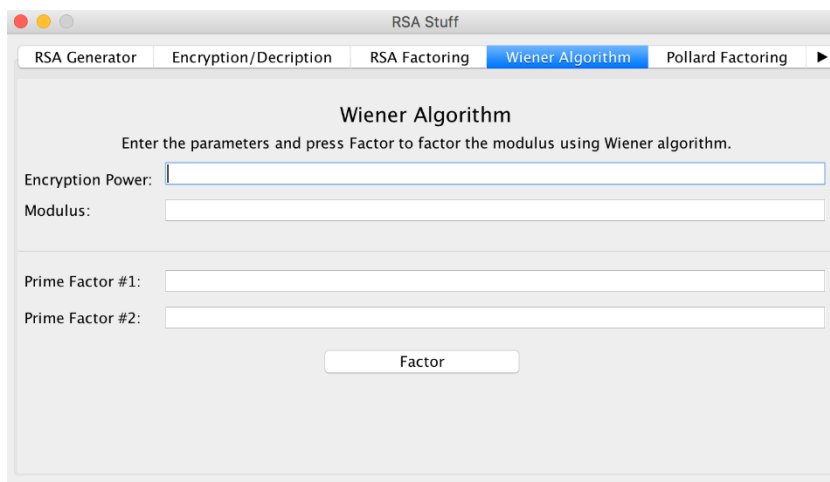
## פירוק המודולוס לגורמים

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  $\phi(n) = (p-1)(q-1)$ . (1) ניתן להגיע לחזקת ההצפנה על ידי  $de = 1 \bmod \phi(n)$ . 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.



The screenshot shows a window titled "RSA Stuff" with several tabs: "RSA Generator", "Encryption/Decryption", "RSA Factoring" (selected), "Wiener Algorithm", and "Pollard Factoring". The "RSA Factoring" tab is active, displaying the title "RSA Factoring" and the instruction "Enter the parameters and press Factor to factor the modulus. You can choose number of cycles in the algorithm." Below this, there is a "Cycles:" input field. Further down, there are input fields for "Encryption Power:", "Decryption Power:", "Modulus:", "Prime Factor #1:", and "Prime Factor #2:". At the bottom center, there is a "Factor" button.

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



The screenshot shows the same "RSA Stuff" window, but with the "Wiener Algorithm" tab selected. The title is "Wiener Algorithm" and the instruction is "Enter the parameters and press Factor to factor the modulus using Wiener algorithm." The input fields are "Encryption Power:", "Modulus:", "Prime Factor #1:", and "Prime Factor #2:". The "Cycles" field is not present in this tab. The "Factor" button is at the bottom center.

איור 6 אלגוריתם וינר לפירוק לגורמים

RSA Generator

Encryption/Decryption

RSA Factoring

Wiener Algorithm

Pollard Factoring

### Pollard p-1 Factoring

Enter the parameters and press Factor to factor the modulus. You can choose the limit B in the algorithm.

Choose B value:

Modulus:

Prime Factor #1:

Prime Factor #2:

Factor

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

## דוגמת הרצה

מצורפת דוגמת הרצה של הצפנת חוברת הקורס עם הפרמטרים הבאים :

The screenshot shows the 'RSA Stuff' application window. It has tabs for 'RSA Generator', 'Encryption/Decryption', 'RSA Factoring', 'Wiener Algorithm', and 'Pollard Factoring'. The 'Encryption/Decryption' tab is active.

**Encryption Section:**

- Encryption Power: 1931706545028527481560013854465438437
- Modulus: 34278715958986161226447219002677245243
- File Address: /Users/guyweissenberg/Desktop/22923.pdf
- Encrypt button

**Decryption Section:**

- Decryption Power: 27219664758601978490313407674841246333
- Modulus: 34278715958986161226447219002677245243
- File Address: /Users/guyweissenberg/Desktop/22923.pdf\_encrypted
- Decrypt button

הקובץ המוצפן זהה לשם הקובץ המקורי בתוספת \_encrypted. בדוגמה, ההצפנה של הקובץ 22923.pdf תהיה הקובץ 22923.pdf\_encrypted. הקובץ המופענח 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):

340928	A4969ECA	FDD8E490	6A483947	92160048	6ED526ED	4379A5E8	ADB80BD4	72545FDF	340896	6C7DE329	30B5DE84	9CE55DA4	CA48D918	679CEE56	25FC3F99	21FE90AC	A984838F
340960	B082041E	080A54F6	033490B0	6122084C	E720C598	3889A3F8	1027E6A3	7817AFE2	340928	A4969ECA	FDD8E490	6A483947	92160048	6ED526ED	4379A5E8	ADB80BD4	72545FDF
340992	93D9705E	C0EBC5E2	47800100	CDBE4956	0D0A656E	64737472	65616000	656E646F	340960	B082041E	080A54F6	033490B0	6122084C	E720C598	3889A3F8	1027E6A3	7817AFE2
341024	626A0031	36392030	206F626A	0D3C3C2F	4465636F	64655061	726D733C	3C2F436F	340992	93D9705E	C0EBC5E2	47800100	CDBE4956	0D0A656E	64737472	65616000	656E646F
341056	6C75606E	7320352F	50726564	6963746F	72203132	3E3E2F46	696C7465	722F466C	341024	626A0031	36392030	206F626A	0D3C3C2F	4465636F	64655061	726D733C	3C2F436F
341088	61746544	65636F64	652F4944	583C4546	39324232	34443930	46444132	30363335	341056	6C75606E	7320352F	50726564	6963746F	72203132	3E3E2F46	696C7465	722F466C
341120	34383934	46413641	38453530	36453E3C	36323930	35424535	39454232	38323446	341088	61746544	65636F64	652F4944	583C4546	39324232	34443930	46444132	30363335
341152	39413739	44423446	38374230	46373936	3E5D2F49	6E666F20	32393420	3020522F	341120	34383934	46413641	38453530	36453E3C	36323930	35424535	39454232	38323446
341184	4C656E67	74682034	39392F52	6F6F7420	32393620	3020522F	53697A65	20323935	341152	39413739	44423446	38374230	46373936	3E5D2F49	6E666F20	32393420	3020522F
341216	2F547970	652F5852	65662F57	58312033	20315D3E	3E737472	65616000	0A68DEEC	341184	4C656E67	74682034	39392F52	6F6F7420	32393620	3020522F	53697A65	20323935
341248	923D4895	5118C7CF	73DE07C4	DB8D7AC3	96401CA5	B49A4A1C	224C2833	50C3A10F	341216	2F547970	652F5852	65662F57	58312033	20315D3E	3E737472	65616000	0A68DEEC
341280	A5948270	710A441C	1CA241CC	866A5042	670723C2	708021A1	0C127480	29280497	341248	923D4895	5118C7CF	73DE07C4	DB8D7AC3	96401CA5	B49A4A1C	224C2833	50C3A10F
341312	401A2A68	02E8EAF3	38C673B8	17C3397A	871F7FFE	CFE739EF	F14E3F2F	6EA0CE9F	341280	A5948270	710A441C	1CA241CC	866A5042	670723C2	708021A1	0C127480	29280497
341344	1D35A2F4	9F95720C	BE87F8C9	39F459CB	9427E819	F4A832FF	5A79A113	A7D7FDCD	341312	401A2A68	02E8EAF3	38C673B8	17C3397A	871F7FFE	CFE739EF	F14E3F2F	6EA0CE9F
341376	77F412F9	38153771	162DA72A	5470B09C	5C8A8E58	8ED3A8C8	02CE03B6	9064B7A1	341344	1D35A2F4	9F95720C	BE87F8C9	39F459CB	9427E819	F4A832FF	5A79A113	A7D7FDCD
341408	F426EC80	ECF5F561	0196C36B	D46ED894	6C3FCEA8	39C923F4	92CD9577	38B33637	341376	77F412F9	38153771	162DA72A	5470B09C	5C8A8E58	8ED3A8C8	02CE03B6	9064B7A1
341440	19A7DB44	34A81EBF	0FFD0336	C21AAAD6	AC5BC53D	3B7B9858	7503DDAD	3C798B2A	341408	F426EC80	ECF5F561	0196C36B	D46ED894	6C3FCEA8	39C923F4	92CD9577	38B33637
341472	6EC31FB6	1CE16E55	254ED8A1	1266E1A4	902EE941	3F47DF87	54C95178	1E5E853F	341440	19A7DB44	34A81EBF	0FFD0336	C21AAAD6	AC5BC53D	3B7B9858	7503DDAD	3C798B2A
341504	A333F2AF	D07C74B7	6590B8F2	5FF187D1	53F010E4	EFF8396A	37AC5B25	1D1CB72A	341472	6EC31FB6	1CE16E55	254ED8A1	1266E1A4	902EE941	3F47DF87	54C95178	1E5E853F
341536	63CAF25F	D1BBFA44	742C7A63	AFCC49E9	9F5C44B3	6D9287D5	F01B1C20	CA490D17	341504	A333F2AF	D07C74B7	6590B8F2	5FF187D1	53F010E4	EFF8396A	37AC5B25	1D1CB72A
341568	9B927F8A	3E5E3445	3261D6B2	4D29CBE1	2C98136E	32ADE544	D3380CA7	BF0E1FDB	341536	63CAF25F	D1BBFA44	742C7A63	AFCC49E9	9F5C44B3	6D9287D5	F01B1C20	CA490D17
341600	DB489E51	7B011215	B6922BEC	F05179A9	49E7BE08	E7E5609C	1E427729	DB5B9467	341568	9B927F8A	3E5E3445	3261D6B2	4D29CBE1	2C98136E	32ADE544	D3380CA7	BF0E1FDB
341632	DAE8CCDF	9706CD3F	7107879C	B3441D51	7F590972	17E7B6B2	F98DE6D7	F3DEA4D8	341600	DB489E51	7B011215	B6922BEC	F05179A9	49E7BE08	E7E5609C	1E427729	DB5B9467
341664	A648BBBD	C64C98F8	903F1276	AEF9F332	656B799D	6E49C414	3AC12914	47631EFC	341632	DAE8CCDF	9706CD3F	7107879C	B3441D51	7F590972	17E7B6B2	F98DE6D7	F3DEA4D8
341696	6B66A624	7A207276	FDDFCAS5	4966696D	BA47E7B4	646E368A	1E29144F	FCFF7F83	341664	A648BBBD	C64C98F8	903F1276	AEF9F332	656B799D	6E49C414	3AC12914	47631EFC
341728	326BDAEF	E32FCBD6	B0000300	SABA6AF9	0D0A656E	64737472	65616000	656E646F	341696	6B66A624	7A207276	FDDFCAS5	4966696D	BA47E7B4	646E368A	1E29144F	FCFF7F83
341760	626A0073	74617274	78726566	0D0A3131	360D0A25	25454F46	0D0A3000	0D0A0000	341728	326BDAEF	E32FCBD6	B0000300	SABA6AF9	0D0A656E	64737472	65616000	656E646F
341792									341760	626A0073	74617274	78726566	0D0A3131	360D0A25	25454F46	0D0A	

1: Delete 6 bytes at offset 0x5371a

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

## רקע תיאורטי

## אלגוריתם מילר רבין

אלגוריתם מילר-רבין הוא אלגוריתם מונטה-קרלו<sup>2</sup> 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  $a$ ,  $1 \leq a \leq n - 1$ 
 $b \leftarrow a^m \bmod n$ 
if  $b \equiv 1 \pmod{n}$ 
  then return ("n is prime")
for  $i \leftarrow 0$  to  $k - 1$ 
  do {
    if  $b \equiv -1 \pmod{n}$ 
      then return ("n is prime")
    else  $b \leftarrow b^2 \bmod n$ 
  }
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  $a^m \not\equiv 1 \pmod{n}$ . נסתכל על  $b$  באלגוריתם – מעלים את  $b$  בריבוע בכל חזרה של הלולאה. מכיוון שהאלגוריתם החזיר ש- $n$  פריק, אז  $a^{2^i m} \not\equiv -1 \pmod{n}$  לכל  $0 \leq i \leq k - 1$ . לפי ההנחה ש- $n$  ראשוני, ממשפט פרמה נובע ש- $a^{2^k m} \equiv 1 \pmod{n}$  (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 ל-1-. אך ערכו לא יכול להיות -1 שכן אז היה מוחזר ש- $n$  פריק. לכן  $a^{2^{k-1} m} \equiv 1 \pmod{n}$ . לכן גם  $a^{2^{k-2} m}$  הוא שורש ריבועי של 1 מודולו  $n$ , ומאותם השיקולים מתקיים ש- $a^{2^{k-2} m} \equiv 1 \pmod{n}$ . נמשיך כך ונקבל ש- $a^m \equiv 1 \pmod{n}$ , 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.

<sup>2</sup> 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 ההסתברות לשגיאה.

**Algorithm 5.8:** POLLARD  $p - 1$  FACTORING ALGORITHM( $n, B$ )

```

 $a \leftarrow 2$ 
for  $j \leftarrow 2$  to  $B$ 
  do  $a \leftarrow a^j \bmod n$ 
 $d \leftarrow \gcd(a - 1, n)$ 
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  $q$ . מכאן ש- $(p-1) | B!$ . לאחר שנעלה את  $a$  בחזקה כך ש- $a \equiv 2^{B!} \bmod n$ , נשים לב שמכיוון ש- $p | n$  חייב להתקיים  $a \equiv 2^{B!} \bmod p$ . מכאן ש- $2^{p-1} \equiv 1 \bmod p$  לפי משפט פרמה, מכיוון ש- $2^{p-1} \equiv 1 \bmod p$ . מכיוון ש- $(p-1) | B!$  הרי שמתקיים  $a \equiv 1 \bmod p$ , ולכן  $p | (a-1)$ . מכיוון ש- $p | n$  הרי ש- $p | d$  עבור  $d = \gcd(a-1, n)$ . כלומר,  $d$  הוא מחלק לא טריויאלי של  $n$  (אלא אם  $a = 1$ ). כיוון של- $n$  יש שני מחלקים לא טריויאליים, נוכל למצוא את השני בקלות.

## פירוק לגורמים של המודולוס בהינתן חזקות ההצפנה והפענוח

אלגוריתם זה הוא אלגוריתם לאס-וגאסי 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  $ab \equiv 1 \pmod{\phi(n)}$ 
write  $ab - 1 = 2^s r$ ,  $r$  odd
choose  $w$  at random such that  $1 \leq w \leq n - 1$ 
 $x \leftarrow \gcd(w, n)$ 
if  $1 < x < n$ 
  then return ( $x$ )
comment:  $x$  is a factor of  $n$ 

 $v \leftarrow w^r \bmod n$ 
if  $v \equiv 1 \pmod{n}$ 
  then return ("failure")
while  $v \not\equiv 1 \pmod{n}$ 
  do  $\begin{cases} v_0 \leftarrow v \\ v \leftarrow v^2 \bmod n \end{cases}$ 
if  $v_0 \equiv -1 \pmod{n}$ 
  then return ("failure")
  else  $\begin{cases} x \leftarrow \gcd(v_0 + 1, n) \\ \text{return } (x) \end{cases}$ 
comment:  $x$  is a factor of  $n$ 

```

איור 10 פירוק לגורמים בהינתן חזקת הפענוח והמפתח הפומבי

<sup>3</sup> 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  $x^2 \equiv 1^2 \pmod{n}$  וגם  $x \not\equiv \pm 1 \pmod{n}$ . בהינתן השורש, נוכל למצוא את המחלקים הלא טריוויאליים של  $n$  על ידי חישוב  $\gcd(x+1, n)$  ו- $\gcd(x-1, n)$ . כלומר, האלגוריתם מוצא שורשים לא טריוויאליים 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:

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

כלומר, אם בייצוג הבינארי של  $n$  יש  $l$  ביטים, האלגוריתם יעבוד אם לחזקת הפענוח יש פחות מ- $1 - \frac{l}{4}$  ביטים בייצוג הבינארי, ו- $q, p$  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( $n, b$ )

```

( $q_1, \dots, q_m; r_m$ )  $\leftarrow$  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  $m$ 
     $c_j \leftarrow q_j c_{j-1} + c_{j-2}$ 
     $d_j \leftarrow q_j d_{j-1} + d_{j-2}$ 
     $n' \leftarrow (d_j b - 1) / c_j$ 
    comment:  $n' = \phi(n)$  if  $c_j / d_j$  is the correct convergent
    do if  $n'$  is an integer
        then  $\left\{ \begin{array}{l} \text{let } p \text{ and } q \text{ be the roots of the equation} \\ \quad x^2 - (n - n' + 1)x + n = 0 \\ \text{if } p \text{ and } q \text{ are positive integers less than } n \\ \quad \text{then return } (p, q) \end{array} \right.$ 
    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- $de \equiv 1 \pmod{\phi(n)}$ , שמתקיים  $de - t\phi(n) = 1$ . בנוסף, מכיוון ש- $n = pq > q^2$ , מתקיים  $\sqrt{n} > q$ .

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

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

מכיוון ש- $d < t < n^{\frac{1}{4}}$  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:  $q_1 + \frac{1}{q_2 + \dots}$ , 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  $p^2 - (n - \phi(n) + 1)p + n = 0$ .

## בבליוגרפיה

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.