Digital Whisper
Technologic Papers

למה RSA טרם נפרץ?

(בגלל החשיבה המתמטית הלא פרקטית)

מאת גדי אלכסנדרוביץ'

הצגתי בבלוג שלי בעבר את שיטת ההצפנה RSA, שהיא ללא ספק אחת משיטות ההצפנה החשובות ביותר בעולם כיום, וגם אתם משתמשים בה ככל הנראה, לכל הפחות בצורה לא מודעת. הכוח של RSA נסמך על בעיה בסיסית בתורת המספרים שטרם נמצא לה פתרון יעיל- פירוק של מספר לגורמים. בהינתן מספר ששווה למכפלת שני מספרים ראשוניים (מספרים שמתחלקים רק ב-1 ובעצמם), $n = pq$, יש למצוא את p ו- q (למשל, בהינתן המספר 221 יש להחזיר 17 ו-13).

למרות מאמצים כבירים שנעשים בתחום, ולמרות כמה אלגוריתמים מתוחכמים שמפרקים לגורמים מספרים ענקיים יחסית מהר, הקרב עדיין אבוד - האלגוריתמים המהירים ביותר הם עדיין לא יעילים מספיק, באופן עקרוני; גם אם הם מצליחים "לאכול" מספרים עד גודל מסויים, הגדלה לא משמעותית של גודל המספרים הללו (לא משמעותית מבחינת זה שעדיין ניתן להשתמש בהם באופן יעיל כדי לבצע הצפנה) הופכת אותם לקשים מדי עבור כל אלגוריתם פירוק לגורמים ידוע. והנה, התברר לי פתאום על ידי חיפוש אקראי באינטרנט, שלמעשה קיימת דרך פשוטה מאוד לפרוץ את RSA שלא שמעתי עליה. הסיבה שאני מקדיש לה מאמר היא שאני סבור שהטעות שב"פתרון" הזה היא בעלת עניין כלשהו בפני עצמה; ושהרעיון שבבסיס הדרך הזו הוא מעניין לכשעצמו וכדאי לפרט עליו. אבל לא אעבוד עליכם - הסיבה האמיתית שבגללה אני כותב את המאמר הזה היא סדין אדום בדמות הטענה "כולם חושבים בצורה מתמטית ולא בצורה פרקטית כמו מתכנתים, ולכן הצפנת RSA עדיין לא נשברה".

ובכן, במה העניין? במאמר הזה שכאמור, נתקלתי בו כמעט במקרה. המאמר נכתב בידי הלמו - בלוגר ישראלי מפורסם ובעל תואר ראשון במדעי המחשב על פי המאמר. המאמר מתחיל בסקירה לא רעה של RSA. הוא מזכיר גם את הסיקור השגוי של גילוי אלגוריתם AKS לבדיקת ראשוניות, שהזכרתי כאן. הסיקור של הלמו קצת נאיבי לטעמי - גם הוא נופל במלכודת ה"כדי למצוא פירוק של מספר צריך לעבור על כל המספרים הקטנים ממנו "עד השורש"", עם האופטימיזציה הבודדת של "לדלג על המספרים בלולאה שאינם ראשוניים" - כל מי שעוסק ולו קצת בתחום יודע שהאלגוריתמים המודרניים לפירוק לגורמים כלל אינם נראים כך; כאמור, הם מתוחכמים הרבה יותר.

אחרי תיאור RSA וכל העניינים הנלווים לכך, מגיע האקשן. תחת הכותרת "איציק חושב בצורה פרקטית", הלמו מציג את איציק המתמטיקאי המתוסכל שעבד כמהנדס תוכנה באחת החברות, ו"הרקע המתמטי היה לו לעזר רב, אבל בהייטק כמו בהייטק, עושים גם דברים פרקטיים עוקפי מתמטיקה", ואז הלמו מציג את הרעיון המרכזי:

> **"כך למשל, בבניית מערכת תוכנה שמצריכה חישובים מהירים מאוד, והמעבד (מיקרופרוסור) בתוך אותה מערכת לא חזק דיו כדי לבצע חישוב מהיר, נעשתה פניה לטבלה בזיכרון שהכילה תוצאות של חישובים מוכנים. הגישה לזיכרון היתה מהירה הרבה יותר מאוסף פעולות חישוב שביצע המעבד, וזו נוצלה על מנת לשפר את העבודה."**

בהמשך הלמו מפרט את הרעיון, ועושה זאת היטב. בבסיסו, הרעיון הוא כזה - נניח שאנחנו רוצים לחשב פונקציה מסובכת כלשהי, שזמן החישוב שלה ארוך. למה להסתבך? במקום שהמעבד יחשב אותה שוב ושוב ושוב, פשוט נשמור בצד טבלה עם כל הקלטים והפלטים האפשריים שלה. למשל, במקום לחשב את $f(x)=x^2$ (נניח שזו פונקציה מסובכת), שומרים טבלה שבה ליד 1 כתוב 1, ליד 2 כתוב 4, ליד 3 כתוב 9 וכן הלאה. כך אנחנו מצמצמים את הבעיה של חישוב הפונקציה לבעיה של ביצוע חיפוש בטבלה. יתר על כן, אם אנחנו בונים את הטבלה כשהיא ממויינת על פי הקלטים, החיפוש יהיה מהיר מאוד - נשתמש בחיפוש בינארי (שעליו סיפרתי ממש לא מזמן) כדי למצוא את הפלט בטבלה, בזמן חיפוש שהוא לוגריתמי בגודל הטבלה (ובעברית - קטן משמעותית מגודל הטבלה). עד כאן - הכל אחלה. השיטה שהלמו מתאר היא אכן שימושית בפרקטיקה, במקרים מסויימים. באשר לתיאוריה המתמטית - עוד נגיע לזה.

ועכשיו אנחנו עולים על הכביש המהיר - "כביש עוקף מתמטיקה לשבירת צופן ה-RSA". הלמו מסביר שהבעיה בצופן היא שבהינתן N, קשה לפרק אותו לגורמיו (למעשה, זה לא מדויק לחלוטין - אולי אפשר לשבור את הצופן גם בלי לפרק את N לא אכנס לכך כעת). גם האלגוריתמים הטובים ביותר שידועים כיום עשויים לקחת זמן רב מדי על קלטים סבירים לחלוטין עבור אלגוריתם ההצפנה. בקיצור, מה שאמרתי בתחילת המאמר. את כל זה הלמו מבטל בהינף יד - "אבל, זו כמובן חשיבה מתמטית ולא חשיבה פרקטית". כמובן. כעת מגיעה הפצצה:

> **איציק טוען שכאשר יוצרים את המספר N, משתמשים בטבלה של מספרים ראשוניים גדולים ידועים, או מחשבים אותם בעזרת אלגוריתם כלשהו. כך למעשה יש בפועל רשימה של כל המספרים הראשוניים בעולם, מ-3 עד p כלשהו. כדי**

לבדוק האם מספר הוא ראשוני בזמן יעיל, אין צורך להפעיל אלגוריתם מתמטי, אלא לחפש את המספר ברשימה שחושבה מראש של כל המספרים הראשוניים הידועים. אם המספר נמצא ברשימה, הרי הוא ראשוני. אם הוא לא ברשימה, אז הוא לא ראשוני.

איציק חצי צודק וחצי טועה. הוא טועה, ובאופן גס למדי, כשהוא טוען שכאשר יוצרים את N משתמשים ב"טבלה של מספרים ראשוניים גדולים ידועים". אני לא מכיר אף אחד שעושה את זה, ומי שעושה את זה עושה דבר מה תמוה ביותר, שכן טבלה שכזו אכן תהיה חשופה להתקפה שאיציק יציע עוד מעט - ואין בכך צורך, שכן יש אלגוריתמים מצויינים למציאת מספרים ראשוניים. איציק מתייחס גם לזה, כמובן, אבל המסקנה שלו שגויה בתכלית, וזו בעצם הטעות המרכזית של המאמר - זה שיש אלגוריתם לחישוב מספרים ראשוניים לא אומר ש"יש בפועל רשימה של כל המספרים הראשוניים בעולם". ממש ממש לא. החלק השני של דברי איציק, שטוען שכדי לבדוק האם מספר הוא ראשוני בזמן יעיל אין צורך בהפעלת אלגוריתם מתמטי ואפשר לחפש אותם ברשימה שחושבה מראש, הוא פשוט שגוי. עוד מעט יתברר למה השיטה הזו שימושית רק עבור קבוצה קטנה מאוד (יחסית) של ראשוניים.

אם כן, מה באמת קורה בעולם האמיתי? כל אחד יכול לקרוא בעצמו; לדוגמא, הספריה OpenSSL שמממשת פרוטוקולי הצפנה אמיתיים זמינה בקוד פתוח לכל ואפשר להציץ בה (כמובן, זה לא אומר שהקוד קריא במיוחד...). למי שמתעניין, הקובץ הרלוונטי הוא bn_prime.c בתת הספריה crypto/bn. בקצרה, הרעיון הבסיסי הוא כזה: מגרילים מספר גדול, בן מספר הספרות המבוקש (איך מבטיחים שמספר יהיה גדול? למשל, כשמגרילים את הביטים שלו מוודאים שהביט המשמעותי ביותר יהיה 1. מן הסתם יש דרכים נוספות). לאחר מכן בודקים שהוא ראשוני - ראשית בדיקת חלוקה נאיבית על אוסף קטן ונתון מראש של ראשוניים (2048 ראשוניים, שפשוט כתובים בטבלה בקובץ bn_prime.h) - עד כאן מזכיר את השיטה של איציק. אלא שכעת, לאחר הבדיקה הנאיבית הזו (שמסננת מספר עצום של מועמדים אקראיים להיות ראשוניים) מורץ אלגוריתם לבדיקת ראשוניות; לא אלגוריתם AKS המפורסם (והאיטי לצרכים פרקטיים), אלא אלגוריתם מילר-רבין ההסתברותי (והמהיר מאוד), שמורץ עם פרמטר בטיחות טוב דיו כדי להבטיח שההסתברות שיתקבל בטעות מספר שאינו ראשוני הוא אפסי. ארחיב על מילר-רבין ועל שיטות אחרות לבדיקת ראשוניות בפעם אחרת; לעת עתה אסתפק בלהגיד שבדומה לבעית הפירוק לגורמים, כך גם השיטה של מילר רבין היא מחוכמת (אם כי לא מתקרבת לרמת התחכום של אלגוריתמי הפירוק לגורמים) ואינה מתבססת על רעיונות נאיביים כמו "בדוק עבור הרבה מספרים אם הם מחלקים את המספר שאת ראשוניותו בודקים".

חזרה אל איציק והרעיון שלו. אחרי שהוא מסביר מהו חיפוש בינארי ולמה הוא יעיל, איציק אומר:

אם היתה קיימת טבלה של כל מספרי ה-N האפשריים שהן כפולות של כל המספרים הראשוניים אחד בשני, לא היה צורך בניסיון להפעיל אלגוריתם מתמטי כדי למצוא את המספרים הראשוניים p ו-q המרכיבים את N.כל שצריך הוא לייצר טבלה כזו. איציק קורא לה "לוח הכפל של המספרים הראשוניים".

איציק צודק לגמרי. טבלה כזו (שבה כתובים ליד המספר 15 הגורמים שלו, 3 ו-5, ליד 21 כתובים 3 ו-7, וכן הלאה) אכן תהיה שימושית מאוד בפירוק לגורמים. כעת איציק נכנס לפרטים הטכניים:

איציק מסביר שהבעיה כיום היא שאין מחשב חזק דיו כדי לבצע חישובים בפרק זמן סביר, אבל זיכרון יש בשפע, וכיום הוא זול מאוד בהשוואה לשנת 1977, השנה בה המציאו את הצפנת ה-.RSA איציק יודע גם להסביר שאם מפתח ההצפנה N הוא למשל בגודל 1024 ביט (שמתאים למספר עשרוני בן יותר מ-300 ספרות), אז כמות הזיכרון שצריך כדי לאחסן את המפתח הוא בסך הכל 128 בתים (מחלקים 1024 ביטים ב-8 כי בכל בית יש 8 ביטים). בנוסף יש צורך ב-128 בתים נוספים על מנת להחזיק את המספר הראשוני p וכן 128 בתים נוספים כדי להחזיק את המספר הראשוני q, שניהם מרכיבים את מפתח ההצפנה N.

החשבון כמובן נכון, אם כי מפתיע אותי שאיציק הפרקטי חושב שצריך לשמור גם את p וגם את q; מספיק לשמור את p ולחשב את q על ידי חלוקת N ב- p . זה מצמצם את גודל הטבלה בשליש. אם כן, הסכמנו שכל כניסה בטבלה היא קטנה מאוד - לוקחת 256 בתים. להשוואה - בעת כתיבת שורות אלו, הקובץ שבו הן נכתבות תופס כבר 25 אלף בתים. אם כן, הכל מושלם - אז למה RSA לא נפרצה? לאיציק הפתרונים:

איציק, מתמטיקאי מתוסכל ותכנת מובטל טוען שכולם חושבים בצורה מתמטית ולא בצורה פרקטית כמו מתכנתים, ולכן הצפנת RSA עדיין לא נשברה. לדעתו של איציק, כשם שילדים בבית הספר היסודי משננים בעל פה את לוח הכפל והחרוצים שבהם יודעים עבור כל מספר בלוח הכפל מי הם גורמיו שהוכפלו אחד בשני (ללא ביצוע פעולה מתמטית כלשהי), כך גם מחשבים יכולים למצוא את הגורמים הראשוניים של המספר N, על ידי פנייה לטבלה מוכנה השוכנת בזיכרון המחשב.

על הטעות שבמאמר אין לי בעיה "לסלוח" - כולם טועים. על הציטוט הזה לא אסלח ולא אשכח. אנסה בקרוב להפריך אותו - קשה לי להסביר עד כמה הוא שגוי מיסודו, אבל אנסה - אבל לפני כן כדאי שאסביר סוף סוף למה איציק טועה ומטעה. ראשית אתן לאיציק לדבר בעד עצמו, ואני מניח שאלו מכם שבקיאים מעט בתחום יזהו את ה"זינוק הקוואנטי" שהוא נוקט בו:

> **תיאור האפשרות לפצח מסרים מוצפנים ב-RSA על ידי חיפוש המפתח הציבורי ב"לוח הכפל של המספרים הראשוניים" יכול לגרום למצב שארגוני ביון של מדינות גדולות ייצרו לוח כפל ענק של כל מכפלות המספרים הראשוניים בעולם, וכל שיצטרכו על מנת לגלות את המפתח הפרטי מתוך המפתח הציבורי הוא לבדוק היכן נמצא המפתח הציבורי בלוח הכפל של המספרים הראשוניים.אם ישנם מליון מספרים ראשוניים ידועים, אז ישנם מליון בחזקת 2 מפתחות ציבוריים אפשריים. מספר הצעדים המקסימלי הנדרשים למצוא את הגורמים הראשוניים הוא פחות מ-40 צעדים (חישוב לוגריתם בבסיס 2 של מליון בחזקת 2).**
>
> **יתרה מזאת: אם ישנם 10 בחזקת 100 מספרים ראשוניים ידועים, אז ישנם 10 בחזקת 200 מפתחות ציבוריים אפשריים. מספר הצעדים המקסימלי הנדרשים למצוא את הגורמים הראשוניים הוא פחות מ-800 צעדים.**
>
> **על פי המיקום של המספר N בלוח הכפל של המספרים הראשוניים, נוכל לחפש ולמצוא ביעילות את הגורמים הראשוניים p ו-q שאותם אין צורך לחפש בעזרת אלגוריתם מתמטי לא יעיל, ובעזרתם לפענח את המסר המוצפן ע"י יצירת מפתח הפענוח (המפתח הפרטי).**

וכאן נגמר המאמר.

לא אתווכח עם טענת ה"פחות מ-800 צעדים" - היא נכונה לגמרי. זו המחשה נהדרת לכוח העצום של החיפוש הבינארי. רק שאלה אחת לי אל איציק - איפה בדיוק תהיה שמורה אותה טבלה אגדית של 10^{200} מפתחות ציבוריים אפשריים?

בואו נעשה לרגע את החשבון של איציק. כבר הסכמנו שכניסה בטבלה דורשת בסך הכל 256 בתים, שזה מספר זעום. כמה בתים לוקחת הטבלה כולה? במקרה הראשון, אמרנו שיש מיליון בחזקת 2 מפתחות ציבוריים אפשריים; כל מפתח שכזה מהווה כניסה בטבלה, ולכן דורש 256 בתים, ולכן בסה"כ נדרשים

$256 \cdot 10^{12}$ בתים. האם זה מספר גדול? ובכן, מדובר בכ-256 טרהבייטים (טרהבייט הוא 1000 ג'יגהבייט; בכל דיסק קשיח סטנדרטי בימינו יש כמה מאות ג'יגהבייטים וכבר מוכרים דיסקים קשיחים של טרהבייט בודד). כלומר, זה מספר קטן ומגוחך עבור סוכנויות הביון.

נפלא; ניתן לשער שגם במקרה השני מקבלים מספר קטן ומגוחך, אז אפשר ללכת לישון בשקט, נכון? הו, לא. אם יש 10^{200} מפתחות אפשריים, התמונה משתנה באופן דרסטי; במקרה הזה צריך $256 \cdot 10^{200}$ בתים כדי לאחסן את כולם. למי שטרם נתקל בחישובים במספרים כאלו זה עשוי להיראות סביר - 200 זה בסך הכל פי 20 מ-12 (אפילו פחות), אז צריך רק להכפיל את המקום פי 20. אלא שזה לא נכון; לא צריך להכפיל את המקום פי 20, אלא פי 10^{188}. המספר הזה הוא עצום. עצום בצורה בלתי נתפסת. אם נניח שכל גרגר חול בעולם (נהיה לארג'ים ונניח שיש 10^{100} כאלו) היה הופך לפתע פתאום לדיסק קשיח של קווינטיליון פטהבייטים (פטהבייט הוא 1000 טרהבייט), כמות הזיכרון שהיה אפשר לאכסן בכולם לא הייתה מתחילה אפילו לגרד את כמות הזכרון שהטבלה של איציק דורשת. במילים פשוטות - אין, לא הייתה אף פעם ולא תהיה אי פעם טבלה בגודל הזה. אם איציק הוא באמת פרקטי כמו שהוא טוען, היה עליו לדעת את זה.

אבל חמור מכך, אם איציק היה מזלזל קצת פחות בתיאורטיקנים ומקשיב להם, אולי הוא היה מבין שאפילו אם בדרך קסומה יצליחו לבנות את הטבלה העצומה הזו, על ידי רתימת כל הכוח של המין האנושי לפרוייקט הזה, זה עדיין לא היה מדגדג לקריפטוגרפים את קצה הזרת; הם פשוט היו מגדילים את המפתח, מ-1024 ביט ל-2048 ביט. תגידו - מה זה משנה? הרי אם קודם נדרשו 256 בתים בשביל כניסה אחת בטבלה, עכשיו יידרשו 512 בתים - בסך הכל הכפלנו את גודל הטבלה פי 2, לא משהו משמעותי. הבעיה כאן היא שאנחנו מתעלמים מהשאלה הבסיסית - מאיפה הראשוניים מגיעים וכמה כאלו יש?

אמרתי קודם שכדי למצוא מספרים ראשוניים פשוט מגרילים מספר גדול ובודקים אם הוא ראשוני. לא אמרתי מה עושים אם הבדיקה נכשלת - פשוט מגרילים מספר חדש ובודקים שוב (או שמשנים קצת את המספר הישן ובודקים שוב). השיטה הזו נשמעת מוזרה קצת במבט ראשון, כי מי מבטיח לנו שניפול אי פעם על ראשוני? אם כמות הראשוניים קטנה יחסית לכמות כל המספרים, אכלנו אותה - נגריל שוב ושוב מספר ולעולם לא ניפול על ראשוני. למרבה המזל, יש יחסית הרבה ראשוניים - זה בדיוק מה שמראה משפט המספרים הראשוניים שהזכרתי בחטף בפוסט הזה. המשפט אומר (בערך) שבין 1 ל- N יש בערך $\frac{n}{\ln n}$ מספרים ראשוניים (וככל ש- N גדול יותר ה"בערך" הזה מדוייק יותר) - זה אומר ההסתברות

להגריל ראשוני בתחום הזה היא $\frac{1}{\ln n}$. מכיוון ש- $\ln n$ הוא בערך מספר הספרות שנדרשות כדי לייצג את N, נובע מכך שכדי להגריל מספר ראשוני בן 100 ספרות צריך בערך 100 נסיונות, כדי להגריל מספר בן 200 ספרות צריך בערך 200 נסיונות, וכן הלאה - מספר הנסיונות הזה הוא קטן מאוד יחסית. זה גם מעיד על כך שמספר הראשוניים הוא גדול מאוד יחסית.

המספר הגדול ביותר שניתן לייצג עם 1024 ביט הוא 2^{1024}; זהו מספר בן 300 ספרות לערך שמן הסתם לא אכתוב פה (מי שרוצה לראותו - שיכתוב 2**1024 בפייתון או רובי). אם כן, כמה ראשוניים בני עד 1024 ביט יש? בערך $\frac{2^{1024}}{1024} = \frac{2^{1024}}{2^{10}} = 2^{1014}$ ראשוניים, וגם 2^{1014} הוא מספר עצום, גם כן בעל בערך 300 ספרות (חדי העין בוודאי שמים לב שאני משתולל כאן עם הבסיס של הלוגריתם על ימין ועל שמאל - לא נורא, מותר לי). בקיצור, כבר ב-1024 ביט יש לנו הרבה יותר ראשוניים ממה שאיציק טוען; הוא מדבר על 10^{100} ראשוניים, ואני אומר שכבר יש 10^{300}, מה שמוביל לטבלה בגודל של 10^{600} - מספר בלתי נתפס.

אבל רגע, מה קורה אם אנחנו עוברים ל-2048 ביט? אז יהיו לנו בערך 2^{2037} ראשוניים (למה?) וזה מספר בן 600 ספרות. הכפלה של מספר הביטים פירושה העלאה בריבוע של כמות הראשוניים הזמינים לנו (שבאה לידי ביטוי בהכפלה של מספר הספרות בתיאור הכמות שלהם). עכשיו הטבלה שלנו תצטרך להיות מגודל 10^{1200} - לא "פי 2" יותר גדולה, אלא פי 10^{600} יותר גדולה מקודם. בניסוח אחר - אם קודם היינו צריכים להשקיע את כל משאבי המין האנושי בטבלה אחת כזו, שכל תא בה מאכסן כמות זעומה של נתונים, כעת נצטרך לבנות מספר עצום של טבלאות שכאלו - טבלה לכל תא בטבלה המקורית. ואם המין האנושי ישיג את ההשיג הכביר והבלתי נתפס הזה, אז הקריפטוגרפים פשוט ימשכו בכתפיים ויעברו ל-4096 ביט, ושוב יעלו בריבוע את כמות הזיכרון הנדרשת. לקריפטוגרפים הגדלות כאלו של אורך המפתח אינן בעיה ממשית, כי כל מה שהן עושות הוא להגדיל פי 2 את הקושי של החישובים המעורבים (הגרלת מספרים, הצפנה וכו'). הגדלה פי 2 היא כאין וכאפס לעומת ההגדלה פי 10^{600} שאיציק זקוק לה. כל זה הוא מקרה פרטי של האבחנה הכללית שבבסיס תורת הסיבוכיות התיאורטית - סיבוכיות אקספוננציאלית (שבה כשמגדילים את הקלט בביט אחד, הסיבוכיות מוכפלת פי 2, ולכן כשמכפילים את גודל הקלט, הסיבוכיות מועלה בריבוע) אינה משהו סביר, באופן כללי (כמובן שבעולם הפרקטי האמיתי, יש דוגמאות נגדיות).

אם כן, הטבלה של איציק איננה רעיון פרקטי; היא הרעיון הכי לא פרקטי שאפשר לחשוב עליו. ועוד לא אמרנו כלום על סוגיית הזמן שצריך כדי לבנות את הטבלה הזו מלכתחילה (ולמען האמת, גם לא ממש צריך). מה בעצם הייתה הטעות של איציק, חוץ מההתעלמות הגסה מהצורך לחשב כמה זכרון, בבתים, צריך בשביל הטבלה של ה- 10^{100} ראשוניים שלו? לדעתי, חוסר ההבנה שלו את הגודל העצום של מרחב הראשוניים שעומדים לרשות הקריפטוגרפים הוא שבלבל אותו. הנקודה היא שלא צריך לבנות "רשימה שחושבה מראש של כל המספרים הראשוניים הידועים" כמו שאיציק תיאר, כדי שניתן יהיה להגריל ראשוניים - המרחב העצום של ה- 10^{300} ראשוניים בני 1024 ביט זמין לנו בלי שנצטרך לאחסן אותו בשום מקום, בזכות אותו "אלגוריתם מתמטי" שאיציק התעקש שאיננו צריכים.

טוב, סיימנו עם זה, אבל אני עדיין רוצה להתייחס לטענת "כולם חושבים בצורה מתמטית ולא בצורה פרקטית כמו מתכנתים", בשתי צורות שונות - פרקטית, ומתמטית. מבחינה פרקטית, כדי לשכנע ש"כולם" דווקא כן חושבים בצורה פרקטית לפעמים (כש"כולם" מתייחס כאן לקריפטוגרפים) אני רוצה לתת דוגמה לתחום בקריפטוגרפיה שהוא מאוד "פרקטי" באופיו - התחום שעוסק ב-Side-Channel Attacks. מכיוון שהוא ראוי למאמר נפרד, רק אגיד באופן כללי מהו - זה התחום שעוסק בתקיפת מערכות הצפנה לא על ידי תקיפת האלגוריתם שבבסיסן באופן תיאורטי, אלא על ידי שימוש במידע נוסף שמופק מכך שהאלגוריתם התיאורטי ממומש במערכת פיזית. דוגמאות לאפיקים שאפשר להפיק מהן מידע הן זמן החישוב שלוקח למעבד לבצע את האלגוריתם, כמות ההספק החשמלי שהוא צורך, החום שהוא פולט ואפילו הרעשים שהוא משמיע (התקפה פרקטית מהסוג האחרון הציע, בין היתר, עדי שמיר, ה-S שב-RSA; באופן כללי שמיר הוא דוגמה טובה למדען מחשב שיכול להיות גם מאוד מתמטי וגם מאוד פרקטי). ההתקפות הללו מזכירות את תעלול "קריאת המחשבות" הבא - אנחנו נותנים למישהו להחזיק מטבע של שקל ביד אחת, ושל חמישה שקלים ביד השניה, ומטרתנו היא לגלות באיזה יד נמצא איזה מטבע. אז אנחנו מבקשים מהמחזיק לכפול את מה שביד ימין ב-14, ואחר כך מבקשים ממנו לכפול את

מה שביד שמאל ב-14, ואז אנחנו מבקשים ממנו לחבר את התוצאות ולהגיד לנו. מן הסתם הוא תמיד יגיד 84, אבל לעתים קרובות נוכל לנחש באיזו יד המטבע על ידי כך שנבחין איזה חישוב לקח לו זמן רב יותר (עם זאת, התעלול הזה יכול להיכשל כל כך בקלות שאף פעם לא העזתי לבצע אותו בעצמי).

כעת להתייחסות השניה, והחשובה יותר - גם מדעני מחשב שחושבים "בצורה מתמטית" יכולים לחשוב על השיטה שאיציק הציע, ולמעשה הם עשו את זה עוד הרבה לפני שאיציק הציע זאת. קיימים מודלים מתמטיים של חישוב שמטפלים בשיטה הזו באופן הרבה יותר כללי - כי מה שאיציק עושה הוא רק המקרה הקיצוני, והלא מעניין, של השיטה. באופן כללי אפשר לחשוב על אלגוריתמים שנעזרים במהלך

החישוב שלהם ב"טבלה" של מידע שחושב מראש (אולי חישוב שדרש זמן רב; ולמעשה, אולי אפילו מידע שלא ניתן לחשב באופן אלגוריתמי - אבל כדי לפרט על זה, אני זקוק למאמר נפרד), והמודל הפורמלי מכונה "מכונות שמקבלות 'עצה'" (ה"עצה" היא אותה טבלה). העובדה שכל פונקציה ניתנת לחישוב בקלות בהינתן עצה שהיא בעצם טבלה שבה לכל קלט כתוב הפלט המתאים לו היא אחת מהאבחנות הטריוויאליות הראשונות של חקר המודל הזה, כמו גם האבחנה שטבלה כזו היא אקספוננציאלית באורכה ולכן העסק לא כל כך מעניין. בדרך כלל עוסקים בטבלאות שגודלן הוא סביר - פולינומי - ביחס לגודל הקלט; מחלקת הסיבוכיות המרכזית בהקשר זה נקראת P/poly (ה-P מייצג חישוב בזמן יעיל, פולינומי; ה-poly מייצג עצות שהן פולינומיות בגודלן). פרט למודל הזה, מדעני מחשב עוסקים באופן כללי בשקלול זמן-מול-זיכרון (tradeoff Space-Time), כלומר בשאלה עד כמה ניתן לקצר את זמן הריצה של חישוב פונקציה מסויימת באמצעות שימוש רב יותר בזיכרון, אך גם על זה לא אפרט כרגע.

אז מה המסקנה מכל זה? רק אחת. לכל מי שמדגדג לו ללעוג למדעני המחשב המתמטיים ה"לא פרקטיים", להגיד שהם מנותקים מהעולם האמיתי ומפספסים את הטריקים שנמצאים להם מתחת לאף ושהיו מחסלים להם את התחום - אנא מכם, בבקשה, נסו לחשוב עוד קצת. אולי איננו טיפשים כפי שאתם חושבים שאנחנו.