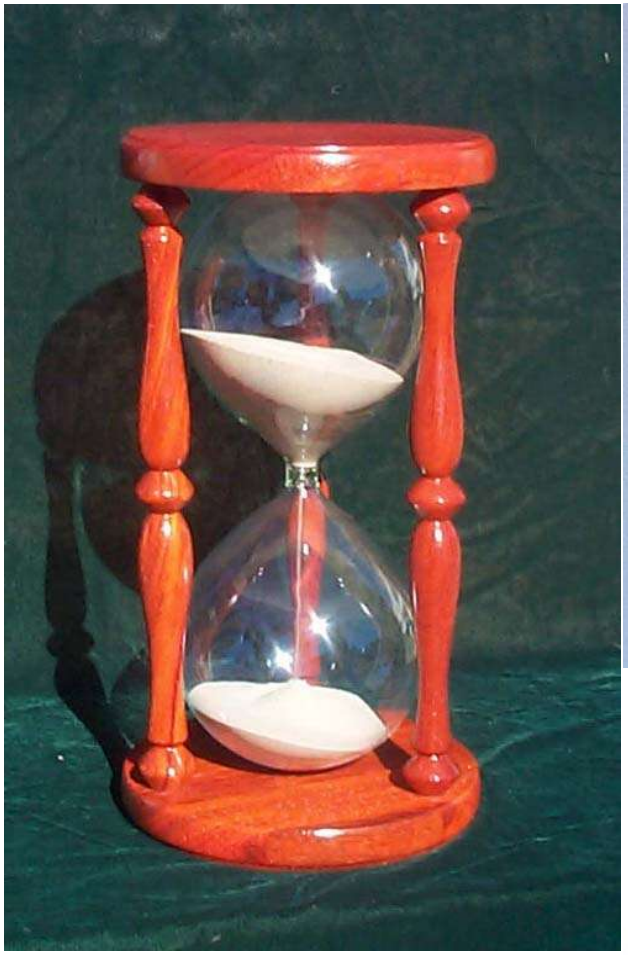
Side Channel Attacks



Orr Dunkelman, Haifux

דלף מידע בזמן חישוב

לכל פעולת חישוב ישנה השפעה על הסביבה:

- שינוי ערכים בזכרון (כולל זכרון מטמון)
- שינוי המצב הפנימי של רכיב החישוב
- פרק הזמן הדרוש לביצוע הפעולה
- צריכת אנרגיה
- קרינה אלקטרומגנטית
- רעש
- שגיאות
- שינויים פיזיים/לוגיים המתרחשים באופן טבעי כתוצאה מהפעולה

דלף מידע בזמן חישוב (המשך)

- תוקף יכול לנצל את המידע הדולף לצורך הסקת מידע על תהליך החישוב
- מידע זה יכול לכלול מפתחות סודים כמו גם תהליכי חישוב "סודיים" (כחלק מתהליך reverse-engineering)

הערות בנוגע להתקפות שנציג

- ההתקפות שנציג בהמשך תלויות במימוש הספיציפי
- עם זאת, מספר המימושים של מרבית הפעולות הקריפטוגרפיות הבסיסיות הוא מצומצם
- תוקף שאיננו מכיר את המימוש, יכול להשיג את המימוש, הן ע"י reverse-engineering, הן ע"י רכישת רכיב דומה, ע"י הצצה בקוד המקור, או אפילו תוך שימוש בהתקפות שנראה בהמשך

התקפות מבוססות זמן חישוב (רקע)

- נתבונן במימוש (הסטנדרטי) הבא של העלאה בחזקה מודולו מספר n:

Input: a, x, n
Output: $a^x \bmod n$

Let $x=x_k x_{k-1}...x_0$
$r = 1$
For $i = k$ downto 0 do:
    $r = r^2 \bmod n$
    if $(x_i=1)$ $r = r*a \bmod n$
output r

התקפות מבוססות זמן חישוב (רקע)

- אם פעולת כפל מודולרית הייתה אורכת זמן קבוע, ניתן בקלות להסיק את מספר האחדים ב-x בהנתן k וחישוב בודד של $a^x \bmod n$
- עם זאת פעולת כפל מודולרית איננה אורכת זמן קבוע – היא תלויה במספרים המוכפלים

Timing Attacks – התקפות תזמון

- הנחת העבודה הבסיסית שלנו היא כי התוקף מסוגל למדוד זמני חישוב וכן כי יש באפשרותו לבחור קלטים לחישוב
- נניח כי ידועים b הביטים העליונים של x, ולכן תוקף יכול לחשב לבד (ולמדוד!) את הזמן הדרוש ל-b האיטרציות הראשונות של האלגוריתם
- אם הערך שהתקבל הוא כזה, שזמן החישוב של האיטרציה הבאה מושפע מאוד מערך הביט ה-b+1 ניתן למדוד את זמן החישוב ולנסות להסיק ממנו את ערך הביט

התקפות תזמון (המשך)

- אם החישוב לרוב לוקח את אותו הזמן, ככל הנראה הפעולה $r*a \bmod n$ לא בוצעה, ולכן $x_{k-b}=0$
- אחרת – $x_{k-b}=1$
- דרך פעולה זו מאפשרת לשחזר את כל x באופן רקורסיבי
- עם זאת, תיתכנה שגיאות במהלך התהליך
- בעצם התהליך המתואר מבוסס על הערכה כמה פעמים התארך החישוב
- ניתן להפוך את הבעיה לבעיית זיהוי אות...

פתרונות להתקפות תזמון

- ניתן להשקיע פעולות סרק כדי למנוע תזמון מדויק
- ניתן לפרק חלק מהפעולות לתתי פעולות אחרות (על פי משפט השאריות הסיני עבור פענוח RSA), כך שהתוקף לא ידע את ה-modulo המשמש בחלוקה
- אפשר לשנות את אופן החישוב (לדוגמא, לטפל בשני ביטים של החזקה בכל צעד)

Power Analysis

- לרוב התקפה זו ממומשת כנגד כרטיסים חכמים ורכיבי מיחשוב דומים
- הרעיון הבסיסי הוא למדוד את צריכת הזרם/מתח של תהליך החישוב ומכן להסיק אינפורמציה
- בדומה להתקפות תזמון – נקבל הערכה לפעולות המתבצעות, לאו דווקא ערך מדויק

דוגמא - Power Analysis

- נתבונן בתיאור הלוגי הבא של מימוש חומרה של שלב AES:

Round Input: *State[16], Subkey[16]*
Output: updated *State[16]*

For *i* = 0 to 15 do:
    State[i]=ByteSub(State[i])
ShiftRows(State)
MixColumns(State)
For *i* = 0 *to 15 do:*
    State[i]^=Subkey[i]

Power Analysis – דוגמא (המשך)

- אם ידוע לנו באיזה רגע מבוצעת פעולת ה-XOR עם המפתח, ניתן לגלות כמה זרם/מתח היא צורכת
- צריכת הזרם של שינוי ביט מצריכת הזרם של השארת המצב על כנו
- צריכת הזרם של שינוי ביט מ-0 ל-1 היא גם גבוהה יותר מצריכת הזרם של שינוי ביט מ-1 ל-0
- לכן, על סמך צריכת הזרם, ניתן לדעת כמה ביטים השתנו, וכמה מהם השתנו מ-0 ל-1
- מידע זה מגלה את ה-Hamming weight של המפתח בשלב, וכן מידע על ערך ה-state

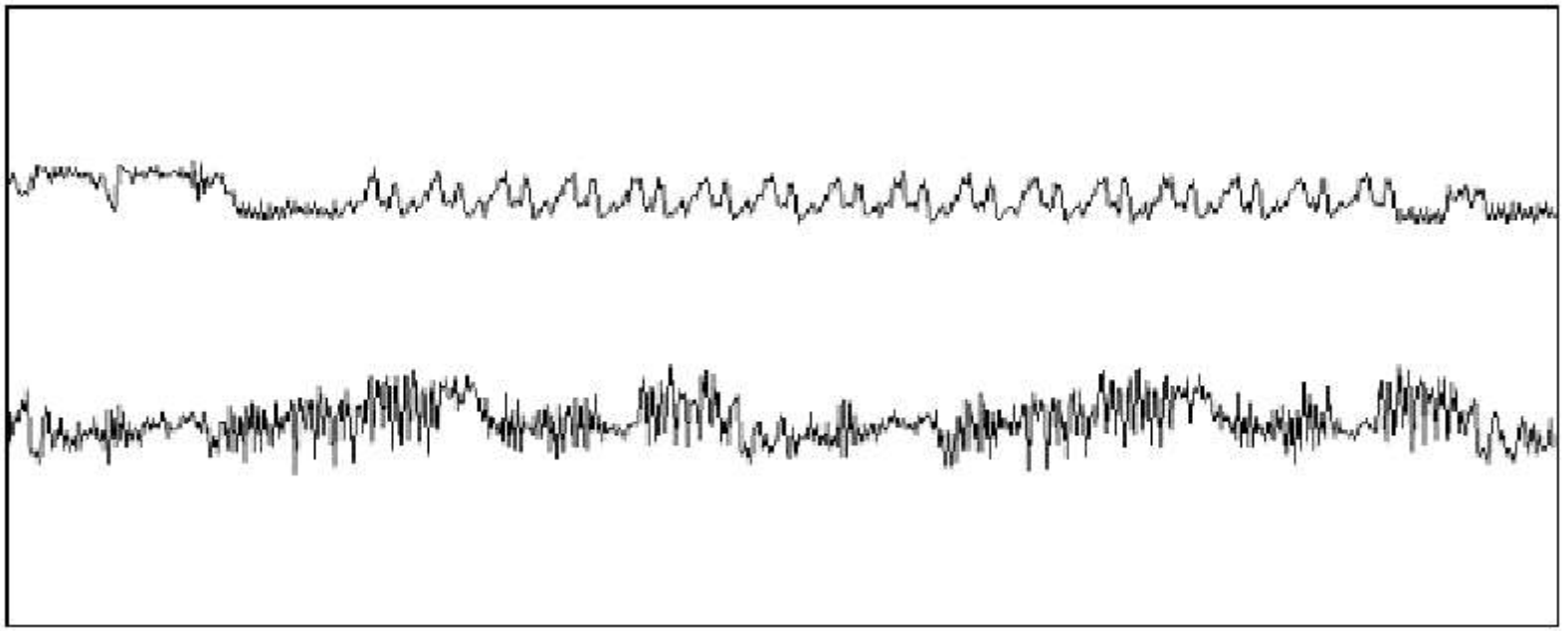
Power Analysis – כיצד התוקף מודד את צריכת הזרם/מתח?

- מרבית רכיבי החומרה שעליהם מפעילים את כלי ה-Power Analysis אינם בעלי מקור אנרגיה פנימי
- לרוב מדובר בכרטיס חכם (smart card) או מעבד המקבלים את הזרם/מתח ממקור חיצוני
- ברוב המקרים של כרטיסים חכמים רבים, מקור הזרם/מתח הוא לא רק חיצוני אלא גם בבעלות התוקף

Simple Power Analysis (SPA)

- ב-SPA התוקף מודד את צריכת הזרם ומנסה להסיק ממנה את הפעולות המתבצעות
- את ההסקה ניתן לבצע על סמך האבחנות הקודמות ודומות להן

Simple Power Analysis (המשך)



- הגרף הנ"ל לקוח ממאמרם של Kocher, Jaffe ו-Jun
- הקו העליון מתאר צריכת זרם של הצפנת DES שלמה (16 שלבים), והקו התחתון הוא הגרף של השלבים השני והשלישי (בהגדלה)

הגנה כנגד Simple Power Analysis

- הגנה כנגד SPA היא "קלה" יחסית
- ניתן ליצור צריכת זרם/מתח קבועה ככל האפשר בציר הזמן
- חישוב הפונקציות בדרכים בלתי שגרתיות
- מיסוך הקלטים והערכים (לדוגמא, שימוש בטבלה ByteSub'(x) = Bytesub(x) XOR 34 וביצוע XOR עם הערך המתאים לאחר מכן)

מיסוך קלטים – המדריך לתוקף המתחיל

- נתבונן בדוגמא הקודמת של שימוש ב
  $$ByteSub'(x) = Bytesub(x)\ XOR\ 34$$
- הצפנה אחת בודדת איננה עוזרת (בהרבה) לתוקף כתוצאה מהמיסוך
- אבל התוקף **איננו מוגבל** להצפנה אחת בודדת

Differential Power Analysis (DPA)

- ניתן להשתמש במספר גדול של הצפנות!
- נתבונן בהצפנת AES:
  - התוקף מבקש 1,000 הצפנות של 1,000 כתבים גלויים
  - התוקף מודד את צריכת האנרגיה בפעולת ה-ByteSub האחרונה של byte מסוים בכל ההצפנות ונחשב לכל זוג (P,C) צריכה ממוצעת
  - לכל ניחוש של byte המפתח הרלוונטי התוקף מבצע פיענוח חלקי של אותו byte
  - עבור ביט מסוים (לדוגמא msb) התוקף מחלק לכאלה בעלי ערך 0 בביט הזה וכאלה עם ערך 1 בביט
  - לכל אחת משתי הקבוצות התוקף מחשב את צריכת הזרם הממוצעת. עבור ערך מפתח נכון, ישנו הפרש ברור בין שתי צריכות הזרם הממוצעת

DPA (המשך)

- את "מיקומן" המשוער של פעולות מסוימות ניתן למצוא תוך שימוש ב-SPA
- ניתן להשתמש באותו סט נתונים עבור מספר גדול של ניסיונות
- ניתן לשלב בהתקפה את האפשרות להחזיר מספר מפתחות אפשריים, כך שהמפתח הנכון נמצא בינהם בסיכוי גבוה יותר (Key Ranking)
- די קשה להגן על מימושי חומרה מהתקפות DPA

Fault Analysis

- גם שגיאות חישוב יכולות למסור אינפורמציה רבה לתוקף
- במיוחד אם יש בידו גם את הערך הנכון
- נתבונן במקרה של RSA בתוך פרוטוקול תקשורת כלשהו
- נניח כי למותקף ישנו מפתח RSA פומבי (n,e), ובמקרה שהפענוח לא "מוצלח" חוזר הערך המפוענח לא נכון לשולח
- מטרת התוקף – מציאת d מתוך השגיאה והמפתח הפומבי

עוד קצת על "פענוח" RSA

- למי שיודע את הפיקטור של $n=pq$ יש אפשרות לחשב את RSA יותר מהר, תוך שימוש במשפט השאריות הסיני (Chinese Reminder Theorem)
- המשפט קובע שכדי לחשב את $a^x \bmod n$ ניתן לחשב את $E=a^x \bmod p$ ואת $F=a^x \bmod q$ וכי קיימים שני קבועים R_1, R_2 כך שמתקיים:

$$a^x \bmod n=(E*R_1*q+F*R_2*p) \bmod n$$

- ניצול המשפט לצורך חישוב מאפשר להאיץ את החישובים

Fault Analysis (המשך)

- נניח כי התרחשה שגיאה בחישוב E אבל לא בחישוב F
- הערך שקיבל המפענח מפענוח c הוא

$$m' = (E' * R_1 * q + F * R_2 * p) \bmod n$$

- בעוד שהערך שנשלח מקיים

$$m = c^d \bmod n = (E * R_1 * q + F * R_2 * p) \bmod n$$

- אם התוקף מקבל גם את ערך הפענוח השגוי הוא מסוגל לחשב את

$$q = gcd(n, m - m')$$

איך שגיאה נולדת?

- ברוב המימושים אין שגיאות חישוב (אחרת לא הייתם קונים אותם)
- לכן צריך "לייצר" שגיאות!
- ע"י הפעלת לחץ פיזי על הרכיב ניתן לגרום לתקלות שונות ומשונות (קריעת ערוץ תקשורת, קיבוע קלט ל-0 או ל-1)
- ע"י מתחים/זרמים לא לפי הנדרש
- ע"י הקרנה אלקטרו-מגנטית

אנליזה אקוסטית

- ניתן גם ל"הקשיב" למעבד/למקלדת
- ע"י אנליזה של הרעש שמייצר המעבד ניתן למצוא את המפתח הפרטי של RSA

- מאמר שפורסם לא מזמן מוצא סיסמאות ע"ס הרעש של הקשות המקלדת

Cache Analysis

- נתבונן במעבד בעל זכרון cache מהיר מאוד, וזכרון איטי
- מימוש "סטנדרטי" של קריאה ל-S-box דוגמת SubBytes הוא מהצורה S[x]
- כחלק מתהליך החישוב החומרה מביאה את השורה (נניח, 64 בתים) הרלוונטית מה-RAM אל ה-cache (אלא אם השורה כבר ב-cache)
- מיקום השורה שהובאה ב-cache נקבע ע"י החומרה, לרוב על סמך מספר ביטים אופיינים ידועים

Cache Analysis (המשך)

- נתבונן בשתי הצפנות של אותו ערך תחת אותו מפתח
- אם באחת מהן מתרחשים יותר cache misses – היא יותר איטית מרעותה (בממוצע)
- רעיון בסיסי:
  - נזהם את זכרון המטמון בשורה אחת
  - אם כתוצאה מזה ההצפנה נעשתה איטית יותר – יש קריאה ל-S-box עם השורה הזו

Cache Analysis (המשך)

- ההתקפה דורשת
  - האפשרות למדוד זמני ריצה של התהליך המצפין
  - רשות לשנות את המטמון של המעבד (משתמש לגיטימי של המערכת)
- שיפור אפשרי
  - לבצע את המדידה על איזו שורה נפגעה בתוך התהליך התוקף
  - כלומר, ל"זהם" את כל המטמון, ולבדוק באילו שורות השתמשה ההצפנה
  - מקל בנושאי מדידת הזמנים

Cache Analysis (המשך)

- במעבדים תומכי Hyper-threading הבעיה רק מחמירה (עקב זיהום הדדי של המטמון)

- על מעבד המממש HyperThreading 64 ההתקפה הבסיסית (עם מספר שיפורים) דורשת כ-300 הצפנות זהות
- הטכניקה השניה מאפשרת להוריד את מספר ההצפנות הדרושות ל-15 בלבד

Cache Analysis של OpenSSL

- Percival מתאר התקפה על פונקציית ההעלאה בחזקה מודולרית של OpenSSL המבוססת על Cache analysis
- ההתקפה הפועלת על עקרונות דומים מסוגלת לשחזר קצת יותר מ-60% מביטי המפתח הסודיים (310 מ-512)
- כתוצאה מההתפקה Ubuntu שיתקו את ההפעלה המיידית של HyperThreading מה-kernel. להפעלה מחדש ht=on בזמן עליה

מידע נוסף

- Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems: http://www.cryptography.com/timingattack
- Kocher, Jaffe, Jun: Introduction to Differential Power Analysis and Related Attacks: http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf
- Boneh, DeMillo, Lipton: On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract), Eurocrypt 1997, pp.37-51.
- Biham, Shamir: Differential Fault Analysis of Secret Key Cryptosystems, Crypto 1997, pp. 513-525.

מידע נוסף (המשך)

- Shamir, Tromer: Acoustic cryptanalysis, On nosy people and noisy machines. http://www.wisdom.weizmann.ac.il/~tromer/acoustic/
- Osvik, Tromer, Shamir: Cache attacks and countermeasures: the case of AES, http://www.wisdom.weizmann.ac.il/~tromer/papers/cache.pdf
- Percival: Cache Missing for Fun and Profit, http://www.daemonology.net/hyperthreading-considered-harmful/