מושגי בסיס בקריפטוגרפיה - מיהם עליזה, בני וחווה?

אור דונקלמן,
מועדון הלינוקס החיפאי, 13/12/2004

קריפטוגרפיה

- מדע ההגנה על מידע
- מטרתה הבסיסית היתה הסתרת מידע, משמע לדאוג שעליזה (Alice) ובני (Bob), בעלי סוד משותף, יוכלו להעביר ביניהם מידע שרק הם יבינו, בלי שחווה (Eve) שמאזינה להם תבין.
- עם הזמן, הורחבה המטרה וכיום היא כוללת גם הגנה כנגד שינוי תוכן המידע, התחזות למקור המידע ועוד...

כל הזכויות שמורות © All rights reserved

פונקציות קריפטוגרפיות

- צפנים:
  - צפני מפתח משותף (סודי, פרטי, סימטרי)
  - צפני מפתח פומבי (אסימטרי)
- חתימות דיגיטלית
- MAC: Message Authentication Codes
- פונקציות תמצות (hash) קריפטוגרפיות
- פרוטוקולים להסכמה על מפתחות
- פרוטוקולי הזדהות קריפטוגרפיים
- הוכחות באפס מידע (Zero Knowledge Proofs)
- חישוב בטוח (Secure Multi-Party Computation)
- פרוטוקולי הצבעה, כסף אלקטרוני, סכימות חלוקת סוד, הצפנה קוונטית, ועוד ...

כל הזכויות שמורות © All rights reserved

צפנים

המטרה: שמירה על סודיות המידע

- שליחת הודעה דרך ערוץ ציבורי כך שרק הגורמים המורשים יוכלו להבין את תוכנה.
- שמירת מידע רגיש (לדוגמא, קוד המקור של חלונות) במחשב במקום לא בטוח (שרת ציבורי), כך שהגורמים הלא מורשים לא יוכלו להבין את המידע.
- שמירת מידע רגיש במחשב, שאמור להשאר בטוח גם לאחר פריצה למחשב.

צופן מורכב מאלגוריתם הצפנה ואלגוריתם פענוח

כל הזכויות שמורות © All rights reserved

צפנים (המשך)

- הרעיון: אחד המשתתפים לוקח טקסט (כתב גלוי – plaintext) אותו הוא רוצה להסתיר, ומבצע עליו פעולת הצפנה. פעולת ההצפנה מתבצעת בעזרת מפתח הצפנה מתאים. לתוצאת ההצפנה קוראים כתב סתר - ciphertext.
  - מי שיש בידיו מפתח הפענוח המתאים, ורק הוא, יכול לגלות מהו הכתב הגלוי מתוך כתב הסתר. זאת נעשה ע"י פעולת הפענוח של כתב הסתר בעזרת מפתח הפענוח.

כל הזכויות שמורות © All rights reserved

צפנים - סימונים

- M – ההודעה הגלויה, Plaintext, כתב גלוי
- C – ההודעה המוצפנת, Ciphertext, כתב סתר
- k_e – מפתח ההצפנה
- k_d – מפתח הפענוח
- E – פונקצית ההצפנה

$$C=E_{k_e}(M)$$

- D – פונקצית הפענוח

$$M=D_{k_d}(C)$$

- חייב להתקיים כי אם k_e ו k_d מפתחות ההצפנה והפענוח המתאימים אזי $M=D_{k_d}(E_{k_e}(M))$

כל הזכויות שמורות © All rights reserved

עקרון קרכהוף

ישנה הנחה קריפטוגרפית (שמופיעה גם בהקשר של אבטחה) – שיטת ההצפנה ידועה. הדבר שאיננו ידוע הוא המפתח שבו השתמשו להצפנה.

ההנחה שהצופן ידוע ופיסת המידע היחידה שחסרה למתקיף היא המפתח שהשתמשו בו להצפנה, מוכרת כ"עקרון קרכהוף".
מערכת שבטוחה גם כאשר השיטה ידועה, בטוחה גם כאשר השיטה איננה ידועה. ההפך <u>איננו</u> נכון!

כל הזכויות שמורות © All rights reserved

צופן מושלם – מפתח חד-פעמי

צופן מפתח חד-פעמי (One Time Pad) הינו צופן מושלם – צופן שאף כוח חישובי אינו מסוגל לפרוץ. אורך המפתח k הוא כאורך ההודעה M שברצוננו להצפין.

פעולת ההצפנה היא ביט אחר ביט, ואנו מבצעים XOR בין ההודעה למפתח. הפענוח הוא ביצוע XOR שוב עם המפתח:

כל הזכויות שמורות © All rights reserved

בטיחות מושלמת וחישובית

- נאמר שצופן הוא Unconditionally Secure (בטוח באופן מושלם), אם בהינתן אינסוף משאבים, ומספר אינסופי של זוגות של הודעות חשופות ומוצפנות, לא ניתן למצוא plaintext בהנתן ה-ciphertext הבא.
- מכיוון שצפנים כאלה אינם מעשיים, לא נדרוש Unconditional Security, אלא נסתפק ב- Computational Security (בטיחות חישובית). דרישה זו היא מעשית יותר.

כל הזכויות שמורות © All rights reserved

בטיחות מושלמת וחישובית (המשך)

בכל האלגוריתמים בהם משתמשים היום (האלגוריתמים ה"מעשיים"), או שאורך המפתח הרבה יותר קטן מאורך ההודעה, או שמצפינים הרבה הודעות עם אותו מפתח.

אם נתון כוח חישוב בלתי מוגבל, ניתן לשבור את כל הצפנים ה"מעשיים", על ידי ביצוע **חיפוש ממצה** (exhaustive search) על מרחב המפתחות.

לכן נדרוש שמספר המפתחות האפשריים יהיה גדול דיו – לדוגמא 2^{128} מפתחות אפשריים.

כל הזכויות שמורות © All rights reserved

בטיחות חישובית

אלגוריתם הצפנה יקרא computationally secure (בטוח חישובית) אם קשה מאד לשחזר את הטקסט המקורי בהינתן הטקסט המוצפן (למשל, הזמן הדרוש לכך ארוך מדי).

לגבי אף אלגוריתם בו משתמשים היום, לא קיימת הוכחה של קושי פיצוח (אומנם חלק מהם מסתמכים על הנחות מתמטיות סבירות).

בהיעדר הוכחה לבטיחות, צופן נחשב חזק אם הרבה זמן לא הצליחו לשבור אותו.

כל הזכויות שמורות © All rights reserved

התקפות על צפנים

בהנתן צופן E שאנו רוצים לתקוף, ניתן לבצע התקפות רבות:

- פענוח כתבי הסתר
- גילוי המפתח שהשתמשו בו להצפנה
- "הצפנת" הודעות נוספות (גם ללא ידיעת המפתח)

ישנם מודלים שונים להתקפה, אך כולם מניחים ידיעה של:

- אלגוריתם ההצפנה
- איפיון סטיסטי של ההודעות הגלויות (לדוגמא – כל ההודעות הן באנגלית). כלומר, תוקף מסוגל לדעת אם הודעה היא "הגיונית"

כל הזכויות שמורות © All rights reserved

התקפות על צפנים (המשך)

- התקפת Ciphertext Only:

לתוקף קבוצת כתבי סתר $\{C\}$.
לרוב התוקף ינסה להשיג את הכתבים הגלויים המתאימים.

- התקפת Known plaintext:

לתוקף קבוצת זוגות של כתבי סתר והכתבים הגלויים שלהם $\{(P,C)\}$.
לרוב התוקף ינסה להשיג את המפתח, או להיות מסוגל להצפין ולפענח הודעות שאינן בקבוצת ההודעות שהוא מכיר.
דוגמא: חיפוש ממצה (Exhaustive Search)

כל הזכויות שמורות © All rights reserved

התקפות על צפנים (המשך)

- התקפת Chosen Plaintext:

התוקף בוחר קבוצת כתבים גלויים $\{P\}$ ומבקש מהמותקף להצפין לו אותם תחת המפתח הלא ידוע. בהנתן קבוצת כתבי הסתר, לרוב התוקף ינסה להשיג את המפתח.
דוגמא: Differential Cryptanalysis

- התקפת Adaptive Chosen Plaintext:

התוקף מבקש הצפנה של קבוצת הודעות, מקבל את כתבי הסתר, מבקש הצפנה של קבוצות הודעות נוספות (על סמך מה שקיבל) וכן הלאה.

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

צפני בלוקים

- צופן בלוקים מקבל קלט בגודל נתון (גודל הבלוק) יחד עם מפתח סודי ומצפין אותו להודעה בגודל נתון (לרוב אותו הגודל).
- בכדי להצפין הודעה M, שאורכה עולה על אורך הקלט של הצופן, מחלקים את M לבלוקים שאורכם כאורך הקלט של הצופן (אם יש צורך מבצעים דיפון). כל בלוק עובר הצפנה בנפרד.
- ניתן להצפין כל בלוק על סמך בלוקים קודמים או באופן בלתי תלוי. הקשר בין הצפנת בלוקים עוקבים נקרא Mode of Operation.

כל הזכויות שמורות © All rights reserved

צפני בלוקים (המשך)

- תהי M הודעת הקלט המחולקת לבלוקים $M=M_1M_2\dots M_n$
- ניתן להצפין כל בלוק באופן בלתי תלוי בבלוקים קודמים על ידי שימוש באותו מפתח k. מוד הפעולה הזה נקרא Elctronic Code Book ומתקיים: $C_i=E_k(M_i)$
- ישנם צפני בלוקים רבים. בין הנפוצים בשימוש: AES,DES,IDEA.

כל הזכויות שמורות © All rights reserved

Data Encryption Standard (DES)

- פותח ע"י I.B.M. בשנות ה-70 אומץ ע"י NIST.
- גודל בלוק – 64 סיביות
- אורך המפתח – 56 סיביות
- התקן היה בשימוש נרחב מאז במשך למעלה מ-25 שנה.

במשך תקופה ארוכה אורך המפתח הקצר היווה נקודה חלשה של DES.
החל משנת 1997 נערכו DES Challenges – תחרויות בהן מפורסם כתב גלוי וכתב סתר מתאים שהוצפנו תחת מפתח לא ידוע. מטרת האתגר – זיהוי המפתח ששימש להצפנה.

כל הזכויות שמורות © All rights reserved

DES Challenges

- האתגר הראשון שוחרר בפברואר 1997. אחרי ארבעה חודשי עבודה (של 78,000 מחשבים דרך האינטרנט) נמצא המפתח.
- האתגר השני שוחרר בינואר 1998. באמצעות קבוצת distributed.net תוך 39 יום נמצא המפתח.
- האתגר השלישי שוחרר ביוני 1998. קבוצת Electronic Frontier Foundation תכננה מכונה בעלות כוללת של 210,000$ (תכנון+בניה בפועל) שמצאה את המפתח המבוקש תוך 56 שעות.
- אתגר נוסף ששוחרר ביולי 1998 נפתר תוך 22 שעות ע"י שיתוף פעולה בין EFF ו-distributed.net.

כל הזכויות שמורות © All rights reserved

Advanced Encryption Standard (AES)

לנוכח אורך המפתח הקצר, וחוסר היעילות של DES הוחלט ב-NIST כי הגיע הזמן להחליפו. ב-1997 פנה הארגון לקהילה הקריפטוגרפית בבקשה להציע הצעות ל-AES.

15 הצעות הוגשו – ומתוכן 5 נבחרו לשלב ניתוח מעמיק:

- MARS
- RC6
- Rijndael
- Serpent
- Twofish

כל הזכויות שמורות © All rights reserved

Rijndael (AES)

בשנת 2000 הוכרז Rijndael כסטדנרט ההצפנה החדש, ובשנת 2001 יצא הסטדנרט באופן סופי.

- גודל הבלוק – 128 סיביות
- אורך המפתח – 128, 192 או 256 סיביות
- יעילות בתוכנה (מימוש על Pentium דורש בערך 20 מחזורי שעון לצורך הצפנת byte)
- יעילות בחומרה

ל-Rijndael מספר סיבובים משתנה (בין 10 ל-14) בהתאם לאורך המפתח. ההתקפה התיאורטית הכי טובה הידועה עליו היא כנגד 7 שלבים.

כל הזכויות שמורות © All rights reserved

צפני שטף (Stream Ciphers)

צפני שטף הינם צפנים המיועדים לשימוש במקרים בהם המידע מגיע בשטף – לדוגמא, שיחת טלפון. במקרה זה מצפינים "בלוקים" קטנים (1,2,4 או 8 סיביות).

באופן עקרוני, צפנים אלה מהירים יותר מצפני בלוקים אך נחשבים לבטוחים הרבה פחות.

ההצפנה מבוססת לא רק על המידע המוצפן כרגע והמפתח, אלא גם על מספר הבלוק, ערכי הבלוק(ים) הקודם(ים), ועוד.

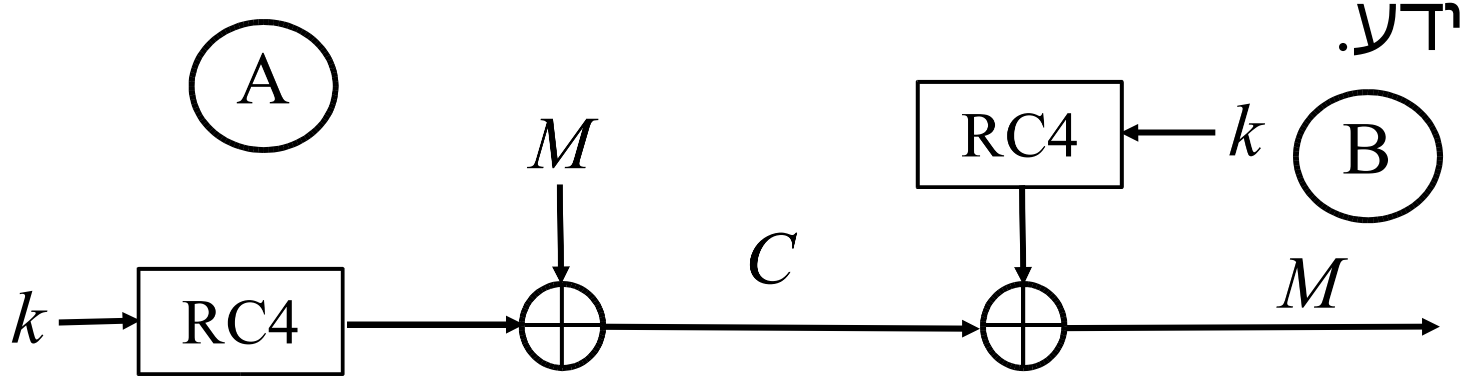
ישנם צפני שטף רבים, אך רובם נפרצו ונחשבים כלא בטוחים. בתהליך דומה לתהליך ה-AES הוצעו כ-7 צפני שטף כהצעה לסטנדרט ע"י טובי מתכנני צפני השטף. שנה וחצי לאחר קבלת ההצעות **כולם** נפרצו.

כל הזכויות שמורות © All rights reserved

דוגמא לצופן שטף - RC4

צופן השטף RC4 נחשב כסטנדרט דה-פקטו. זאת על אף העובדה שיש לו בעיות בטיחות רבות, והיותו של הצופן סוד מסחרי.

בהנתן מפתח k (באורך של עד 256 בתים), הצופן עובר איתחול, ולאחריו פולט רצף סיביות פסואדו-אקראי המשמש מפתח למעין One Time Pad.
הצפנה (ופענוח) – ביצוע XOR בין המפתח שיצא והמידע.



כל הזכויות שמורות © All rights reserved

צפני מפתח פומבי

צפני מפתח פומבי

- לכל משתמש יש זוג מפתחות:
  - מפתח פומבי המשמש להצפנה
  - מפתח פרטי המשמש לפענוח והידוע רק למשתמש
- בהנתן מפתח ההצפנה קשה (חישובית) למצוא את מפתח הפענוח
- רוב הצפנים הללו מבוססים על ההנחה שקיימות בעיות מתמטיות שקל מאוד לפתור בהנתן מידע מסוים וקשה אחרת (trapdoor functions). לדוגמא, שיטת RSA שנציג מיד מבוססת על בעיית פירוק לגורמים ראשוניים של מספר פריק גדול (factoring)

כל הזכויות שמורות © All rights reserved

צפני מפתח פומבי (המשך)

צפני מפתח פומבי הם מעין תיבת דואר מאובטחת:
כל אחד יכול לשלשל מעטפה אל התיבה, ורק בעליה של התיבה מסוגל למצוא מה נשלח אליו.

במציאות, כדי להשיג תכונות בטיחות חזקות יותר, משתמשים בצפני מפתח פומבי באופן די שונה ממה שנראה היום. עם זאת, העקרונות הם זהים, ומשתמשים באותן פונקציות הצפנה שנראה.

כל הזכויות שמורות © All rights reserved

RSA

RSA ראשי התיבות של ממציאיו Rivest, Shamir ו-Adelman, הינו צופן המפתח הפומבי הראשון שהוצע (שנת 1978). מפתחיו כובדו בפרס טיורינג בשנת 2004 על הפיתוח.

המפתח הפומבי של המשתמש הוא זוג מספרים שלמים (n,e), כאשר n הינו המודולו בו מתבצעים החישובים ו-e הוא המעריך (אקספוננט) הפומבי המשמש להצפנה:

$$c=m^e \pmod{n}$$

עבור הודעה m המקיימת $n>m>0$

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

RSA

המפתח הפרטי של המשתמש הוא מספר שלם, המכונה המעריך הפרטי – d. את ערך המודולו ניתן לקחת מהמפתח הפומבי או להתייחס אליו כחלק מהמפתח הפרטי. עבור כתב סתר c המקיים $n>c>0$ הפענוח הינו:

$$m=c^d \pmod n$$

כל הזכויות שמורות © All rights reserved

דרישות מ-RSA

ברור שחייבים להתקיים כל מיני קשרים בין n,e ו-d. נרצה שיתקיימו התנאים הבאים:

- פענוח הודעה מוצפנת ייתן את ההודעה המקורית, כלומר שיתקיים
  $$(m^e)^d \ (\mathrm{mod}\ n) = m$$
- בהנתן n,e קשה למצוא את d.
- המשתמש החוקי (בעל המפתח) כן יהיה מסוגל למצוא את d.

כל הזכויות שמורות © All rights reserved

RSA

יצירת מפתחות:

- בחר זוג ראשוניים גדולים (512 סיביות או יותר) p,q.
- מצא זוג מספרים e,d כך ש-$ed=1(\mathrm{mod}\ (p-1)(q-1))$.
- פרסם את (n,e) כמפתח פומבי ושמור את d כפרטי.

הצפנה של הודעה m נעשית ע"י העלאת ההודעה בחזקת e מודולו n:

$$c=m^e\ (\mathrm{mod}\ n)$$

הפענוח של כתב סתר c נעשה ע"י העלאת כתב הסתר בחזקת d מודולו n:

$$m=c^d\ (\mathrm{mod}\ n)$$

כמובן שמתקיים:

$$D(E(M))=D(m^e\ (\mathrm{mod\ n}))=m^{ed}\ (\mathrm{mod\ n})=m$$
$$E(D(M))=E(m^d\ (\mathrm{mod\ n}))=m^{ed}\ (\mathrm{mod\ n})=m$$

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

בטיחות RSA

הקושי של RSA מסתמך על הקושי של מציאת d ללא ידיעת הפירוק לגורמים ראשוניים של n.
בעיה זו נקראת בעית RSA:
קלט: (e,n)
פלט: d המקיים $ed=1(\mathrm{mod}\ (p-1)(q-1))$.

ברור כי מי שיודע לפתור את בעית הפירוק לגורמים ראשונים (factoring) מסוגל לפתור גם את הבעיה הזו, אם כי יתכן שניתן לשבור את RSA גם ללא פירוק.

בעית הפירוק נחקרת כבר שנים (עוד לפני שהקריפטוגרפים גזלו אותה מאנשי המחקר התיאורטי של תורת המספרים), אך עד היום הבעיה נחשבת קשה.

כל הזכויות שמורות © All rights reserved

פירוק לגורמים – מאז ועד היום

- RSA-129 הינו מספר בן 129 ספרות (426 סיביות), אשר פורסם ב-1977.
- פורק לגורמים ב-1994 ע"י שימוש ב-1600 מחשבים (דרך ה-Internet), תוך שמונה חודשים.
- היום, אלגוריתם הפירוק הטוב ביותר יכול לפרק מספר באורך 512 סיביות במספר ימים תוך שימוש בעשרות מחשבים וחומרה ייעודית בעלות משוערכת של 10 מיליון דולר.
- כיום, באופן מעשי, לא ניתן לפרק לגורמים ראשוניים מספרים בני 1024 סיביות או יותר.

כל הזכויות שמורות © All rights reserved

צפנים – הערה לסיכום

צפנים נועדו להבטיח את **סודיות המידע** בלבד!
הם אינם מבטיחים את שלמותו של המידע.

לדוגמא, בצופן RC4 החלפת ביט בכתב הסתר
מחליפה את ערך הביט בכתב המפוענח. במיוחד
כאשר ידוע לנו חלק מהמידע המוצפן, יש לנו
יכולת לשלוט במה יפוענח.

לפיכך חשוב להשתמש באמצעים המבטיחים
שלמות, דוגמת MAC וחתימה דיגיטלית, שיבטיחו
שהמידע לא שונה.

כל הזכויות שמורות © All rights reserved

אימות ושלמות -
חתימות דיגטליות ו-MAC

מהו אימות?

אימות הוא תהליך באמצעותו ניתן לוודא בנוגע למידע מסוים את:

- זהות השולח (Identity)
- שלמות המידע (Integrity)

פירוש הדבר הוא כי המידע הגיע ממקום ידוע, ולא שונה בדרך.

כל הזכויות שמורות © All rights reserved

אימות באמצעים קריפטוגרפיים

קיימים שני סוגים של פרימטיביים קריפטוגרפיים לביצוע אימות:

- מבוססי מפתחות פומביים – חתימות דיגטליות (אלקטרוניות) Digital Signatures.
- מבוססי מפתח משותף – Message Authentication Codes.

שני סוגי הפרימטיביים מספקים אימות מקור השולח, כאשר לרוב חתימות דיגטליות מספקות גם הגנה על שלמות ההודעה (MAC תמיד שומר על שלמות).
תכונה נוספת שיש לחתימות דיגטליות (ואין ל-MAC) היא אי-הכחשה (Non-repudiation).

כל הזכויות שמורות © All rights reserved

אלגוריתמי חתימה דיגטלית

חתימה דיגטלית מבוססת על מפתח פומבי. לפני שמשתמש U חותם על מסמכים הוא מפרסם את Pub_U המפתח הפומבי שלו. במקביל הוא שומר אצלו את המפתח הפרטי שלו Pr_U.

שיטת חתימה דיגטלית היא בעצם זוג אלגוריתמים:

- אלגוריתם חתימה – $sig=S(M, Pr_U)$ כאשר sig היא החתימה.
- אלגוריתם בדיקה – $V(M,sig,Pub_U)$=true/false האלגוריתם מחזיר true אם החתימה sig על הודעה M נוצרה ע"י U או false אחרת.

זוג (M,sig) עבורו אלגוריתם הבדיקה מחזיר true נקרא זוג חוקי (או לגיטימי).

כל הזכויות שמורות © All rights reserved

דרישות מאלגוריתמי חתימה דיגיטלית

דרישות מאלגוריתם החתימה:

- בהנתן M ו-Pr_U קל לחשב את $S(M,\mathrm{Pr}_U)$
- בהנתן Pub_U קשה לחשב מתוכו את Pr_U
- ללא ידיעת Pr_U קשה למצוא זוגות (M,sig) כך ש-$V(M,sig,Pub_U)=\mathrm{true}$
- בהנתן סט גדול של זוגות $\{(M,sig)\}$ ו-Pub_U קשה לייצר זוג (M,sig) חוקי שאיננו בסט

הדרישות הנ"ל מציגות את מה שאנו דורשים במציאות מחתימות - אדם מסוים U הוא היחיד המסוגל לחתום בשמו. אדם אחר לא יצליח לזייף חתימה של U, גם אם יש לו הרבה חתימות קיימות של האדם.

All rights reserved © כל הזכויות שמורות

חתימה דיגטלית - המשך

נהוג כי ההודעה החתומה היא שרשור ההודעה המקורית m והחתימה sig, כלומר $m||sig$.

לנוכח הדרישות הנ"ל, חתימה דיגטלית מספקת:

- אותנטיקציה (אימות) זהות חותם ההודעה, מכיוון שהוא היחידי שמסוגל לייצר חתימה.
- אי הכחשה - מכיוון שהחותם הוא היחיד המסוגל לייצר את החתימה (הוא הבעלים היחיד של Pr_U), קיום חתימה חוקית מעיד כי הוא חתם על ההודעה.
- שלמות – שינוי ההודעה מחייב שינוי החתימה (עקב הדרישות הנ"ל).

כל הזכויות שמורות © All rights reserved

חתימה דיגטלית - המשך

ישנן מספר רב של שיטות חתימה דיגטלית. לכל אחת מהן בסיס מתמטי שונה, והנחות אחרות. כמו עבור צפני מפתח פומבי, אין לאף שיטת חתימה הוכחה מלאה כי היא עומדת בדרישות שהזכרנו.

בין השיטות הנפוצות והידועות:

- El-Gamal: מבוססת על בעיית הלוגריתם הדיסקרטי שנראה בהמשך. אורך החתימה הוא בד"כ 1024 סיביות.
- Digital Signature Algorithm (DSA): עיבוד של NIST לשיטה של El-Gamal. אורך החתימה – 320 סיביות.
- ECDSA: גרסא של ה-DSA המבוססת על עקומות אלפיטיות.על אף אורך החתימה הזהה של 320 סיביות, השיטה נחשבת הרבה יותר בטוחה מ-DSA.

מועדון הלינוקס החיפאי
All rights reserved © כל הזכויות שמורות

חתימה דיגלית - דוגמא

לפני הדוגמא עצמה, נקדים ונזהיר שזוהי דוגמא בעייתית. השימוש ב-RSA לצורך חתימות הוא טריקי, ויש לעשותו תוך שימוש במנגנונים הנכונים (שהם הרבה יותר מורכבים ממה שנראה כעת).

באתחול, המשתמש בוחר לעצמו מפתח RSA פומבי כמו בהצפנה – (n,e) ושומר לעצמו את המפתח הפרטי המתאים d.

לצורך חתימה על הודעה m, המשתמש מחשב

$$sig=m^d \pmod n$$

הבדיקה של הודעה חתומה (m,sig) היא האם

$$m=sig^e \pmod n$$

כל הזכויות שמורות © All rights reserved

חתימה דיגטלית – המשך דוגמא

מכיוון שרק המשתמש יודע את d הרי שהוא היחידי שיכול לחתום.

המפתח הציבורי (n,e) ידוע לכל העולם, שמסוגל לאמת את החתימה.

ניתן להראות כי מי שמסוגל לייצר חתימה על הודעה נתונה m מסוגל גם לפענח הודעות שהוצפנו תחת המפתח הפומבי של המשתמש.

עם זאת יש לחתימות בשיטה הנ"ל מספר בעיות עיקריות. החמורה ביותר בינהן היא היכולת לייצר זוגות הודעה-חתימה "לגיטמיות" ללא ידיעת המפתח הפרטי.

ישנן דרכים שונות לשימוש ב-RSA כבסיס לשיטת חתימה, כאשר הנפוצה והבטוחה בין השיטות נקראת RSA-PSS.

כל הזכויות שמורות © All rights reserved

בעיות של פונקציות חתימה דיגיטליות

- שימוש בחתימה דיגיטלית מגדיל את גודל הנתונים שיש להעביר
- זמן חישוב החתימה הדיגיטלית ארוך מאוד ותלוי באורך ההודעה (זוהי קריפטוגרפיית מפתח פומבי)
- זמן הבדיקה של החתימה אף הוא ארוך מאוד ותלוי באורך ההודעה

פתרון : במקום לחתום על ההודעה המקורית, חותמים על טביעת אצבע (Digest) של ההודעה (שהיא הרבה יותר קצרה מההודעה המקורית).

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

יצירת "טביעת אצבע"

נרצה כי יצירת טביעת האצבע של ההודעה תהיה בטוחה. השאלה היא מה המובן של בטוח בהקשר שלנו.

דרישות:

- בהנתן הודעה עם "טביעת אצבע" קשה למצוא הודעה נוספת בעלת אותה "טביעת אצבע"
- קשה למצוא שתי הודעות בעלות אותה "טביעת אצבע"

כל הזכויות שמורות © All rights reserved

פונקציות חד כיווניות

פונקציה $y=f(x)$ היא חד-כיוונית, אם:

- f קלה לחישוב: בהנתן x, קל לחשב את $y=f(\mathrm{x})$
- f קשה להיפוך: בהינתן y, קשה למצוא x שמקיים: $y=f(x)$

שימו לב כי פירוש הביטוי "קשה להיפוך" הוא שקשה למצוא מקור x' כלשהו עבור ערך y נתון (ולאו דווקא את ערך x המקורי).

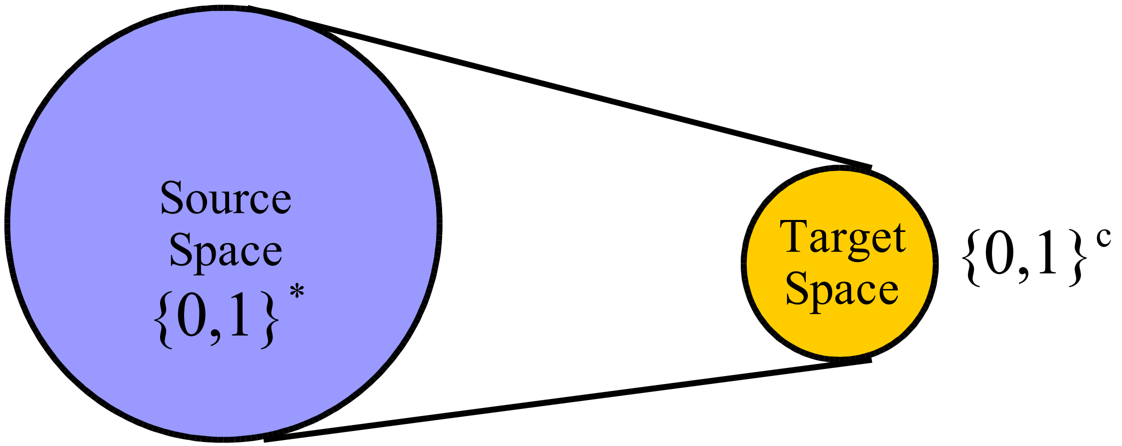
קיומן של פונקציות חד-כיווניות הינה שאלה פתוחה. ידוע כי אם קיימות פונקציות כאלו אזי $P \neq NP$. ההפך אינו בהכרח נכון.

כל הזכויות שמורות © All rights reserved

פונקציות תמצות (hash) קריפטוגרפיות

פונקציה $h(x)$ תקרא פונקציית תמצות קריפטוגרפית אם היא מקיימת את התכונות הבאות:

- כיווץ: אורך הפלט הינו קבוע (על אף שאורך הקלט משתנה)
- חד כיווניות (One Way)
- חסרת התנגשות (Collision Free): קשה למצוא שתי הודעות בעלות אותו digest.



All rights reserved © כל הזכויות שמורות

התקפות על פונקציות תמצות

ישנן שלוש התקפות אפשריות על פונקציות תמצות:

- מציאת התנגשות – מציאת זוג הודעות M1 ו-M2 כך ש-h(M1)=h(M2)
- מציאת התנגשות שניה – בהנתן הודעה M מוצאים הודעה 'M כך ש-h(M)=h(M')
- מציאת מקור – בהנתן y צריך למצוא M כך ש-h(M)=y

ההתקפות שהתפרסמו בקיץ האחרון מצאו התנגשות בפונקציות התמצות MD5 ו-SHA-0.

מועדון הלינוקס החיפאי
All rights reserved © כל הזכויות שמורות

חתימה דיגטלית עם פונק' תמצות

כעקרון, אין שינוי בהגדרות הפורמליות שהשתמשנו בהן קודם. הרי ניתן להניח שאלגוריתם החתימה לוקח את ההודעה M מפעיל עליה את פונקציית התמצות $h(x)$, ולאחר מכן חותם תוך שימוש בחתימה כרגיל.
הבדיקה גם היא לא משתנה באופן עקרוני. כחלק מאלגוריתם הבדיקה אנו מחשבים את $h(M)$ וממשיכים כרגיל.

עם זאת, מטעמי נוחות, נכתוב את הפונקציות הנ"ל בצורה מפורשת. נגדיר את S' ואת V' שמשתמשות ב-S וב-V ובפונקציית תמצות $h(x)$:

$$S'(M,Pr_U) = S\,(h(M),Pr_U) = sig$$
$$V'(M,sig,Pub_U) = V(h(M),sig,Pub_U)$$

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

תכונות של פונקציות תמצות בחתימות

נרצה שהוספת פונקציית התמצות לא תפגע בבטיחות החתימה. כלומר, שעדיין יהיה קשה לייצר זוגות של הודעות וחתימות שיעברו את אלגוריתם הבדיקה, ללא ידיעת המפתח הסודי Pr_U.

הבעיה העיקרית היא שפונקציות תמצות עלולות להכניס התנגשויות לחתימה. זוג הודעות (M_1,M_2) יקרא מתנגש אם $h(M_1)=h(M_2)$. במקרה שכזה, חתימה לגיטמית Sig על הודעה M_1 היא גם חתימה לגיטמית על M_2.

שימו לב כי דרשנו שקשה יהיה למצוא התנגשויות, אך בהכרח יש כאלה (עקרון שובך היונים).

כל הזכויות שמורות © All rights reserved

פרדוקס יום ההולדת

- מהו מספר האנשים שצריכים להיות בחדר בכדי שבסיכוי גדול מ-0.5, יהיה בחדר איש שיום ההולדת שלו נופל בתאריך זהה לשלי? 183.
- מה מספר האנשים שצריכים להיות בחדר בכדי שבסיכוי גדול מ-0.5, יהיו ביניהם שניים שיום ההולדת שלהם באותו תאריך? 23.

דרך פשוטה אחת להצדיק את ה"פרדוקס" היא בכך שיש 253=23*22)/2) זוגות של אנשים בין 23 האנשים, ולכן ההסתברות של זוג מסוים להיות בעל אותו יום הולדת, גדולה מחצי.

כל הזכויות שמורות © All rights reserved

פרדוקס יום ההולדת ופונקציות תמצות

כאשר חותמים על הודעה m, חותמים בעצם על כל ההודעות m' עבורן $h(m)=h(m')$.

פונקציות תמצות קריפטוגרפיות צריכות לקיים את התכונה הבאה: בהינתן הודעה m, קשה מעשית למצוא הודעה m' כך ש-$h(m)=h(m')$.

אבל: הרבה יותר קל למצוא מראש שתי הודעות m_1 ו-m_2, כך ש-$h(m_1)=h(m_2)$.

כל פונקציית תמצות קריפטוגרפית חשופה להתקפת יום ההולדת.

כל הזכויות שמורות © All rights reserved

מסקנה

מכיוון שפרדוקס יום ההולדת משפיע על כל פונקציית תמצות שהיא, יש צורך שהפרדוקס לא יפגע בבטיחות.

כדי למצוא התנגשות כאשר אורך הפלט של הפונקציה הוא n סיביות, צריך לנסות בערך $2^{n/2}$ הפעלות של פונקציית התמצות.

אם נרצה שתוקף יצטרך להשקיע בסביבות 2^{64} הפעלות של פונקציית התמצות בשביל למצוא התנגשות, הרי שגודל ה-digest שהפונקציה צריכה לפלוט הוא 128 סיביות.

כל הזכויות שמורות © All rights reserved

דוגמאות לפונקציות תמצות

- MD4 – הייתה נפוצה בעבר, עד שנמצאה התנגשות. פלט – 128 סיביות.
- MD5 – שיפור של MD4, סטנדרט באינטרנט. פלט – 128 סיביות – נמצאה התנגשות לא מזמן.
- SHA-1 – שיפור של MD4. נוצרה ע"י NSA עבור NIST לשמש כסטנדרט בארה"ב. פלט – 160 סיביות – בטיחותה מוטלת בספק.
- Tiger – פלט – 192 סיביות.
- SHA-224, SHA-256, SHA-384, SHA-512 – סטנדרטים חדשים בארה"ב. פלטיהן באורכים המצויינים (224,256,384,512) סיביות בהתאמה.

כל הזכויות שמורות © All rights reserved

Message Authentication Codes (MAC)

הרעיון: בדומה להצפנה בצופן סימטרי, נניח כי לשני הצדדים B, A שמשתתפים בפרוטוקול יש מפתח סודי k משותף. כדי לאמת שהודעה m הגיעה מאחד מהם, הם יחשבו פונקצייה מסויימת, התלויה ב-k.
כלומר $MAC=f(m,k)$.

דרישות מן ה-MAC:

- ללא ידיעת k לא ניתן לייצר MAC חוקי על הודעה m.
- גם אם לתוקף יש סט של הודעות ו-MAC-ים תואמים, הוא איננו מסוגל לייצר זוג חדש של הודעה ו-MAC שאיננו בסט.
- לא ניתן לשחזר את המפתח k מתוך הודעה m וה-MAC שלה (וגם לא מסט גדול של הודעות ו-MAC).

כל הזכויות שמורות © All rights reserved

Message Authentication Codes (MAC)

- אם A מקבל הודעה יחד עם ה-MAC אותו B חישב על ההודעה בעזרת המפתח הסודי, והאימות של ה-MAC מצליח, A יודע בוודאות שההודעה נשלחה מ-B, ושהיא לא שונתה בדרך.

- בניגוד לחתימה דיגיטלית, במקרה זה A לא יכול לטעון בבית משפט ש-B שלח לו את ההודעה, מכיוון ש-A עצמו גם מסוגל לייצר את ה-MAC.

לכן MAC לא מספק את תכונת אי ההכחשה.

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

MAC - המשך

כלומר MAC הוא פרימטיב סימטרי (משמע מבוסס על מפתח סודי), שמספק שתי תכונות:

- אימות
- שלמות

MAC איננו מספק Non-repudiation (אי הכחשה).

קיימים שלושה סוגים של MACs:

- מבוססי פונקציית תמצות
- מבוססי צופן בלוקים
- מבוססי צופן שטף

כל הזכויות שמורות © All rights reserved

MAC מבוסס פונקציות תמצות

הרעיון: מתמצתים את ההודעה m והמפתח הסודי k ביחד.

דוגמאות (h היא פונקציית תמצות):

Keyed MAC : MAC = $h(k,m,k)$

HMAC : MAC = $h(h \text{ XOR } opad, h(k \text{ XOR } ipad, m))$

$ipad$ ו-$opad$ הם קבועים של HMAC.

תכונות:

- מספק שלמות (עפ"י הגדרת פונקציית תמצות קריפטוגרפית, קשה למצוא שתי הודעות שתתמפנה אל אותו הערך).
- מבצע זיהוי משתמש (ע"י הוספת מפתח סודי לפונקציית התמצות).

כל הזכויות שמורות © All rights reserved

MAC מבוסס צפני בלוקים

הרעיון הוא לנצל את העובדה שללא ידיעת המפתח לא ניתן להצפין מידע.

דוגמא נפוצה ל-MAC מבוסס על צופן בלוקים, הוא CBC-MAC. בשיטה זו הבלוק האחרון של הצפנה ב- CBC mode של ההודעה משמש כ-MAC. כלומר השולח מצפין את ההודעה m תחת המפתח המשותף k תוך שימוש בצופן בלוקים מוסכם וידוע במוד פעולה של CBC. הבלוק האחרון המתקבל בהצפנה – נשלח לצד השני, כעדות לשלמות המידע.

ל-CBC-MAC יש מספר בעיות אבטחה, המאפשרות לייצר הודעה עם MAC גם ללא ידיעת המפתח. עם זאת, הידיעה היא חסרת משמעות (בסיכוי גבוה).

כל הזכויות שמורות © All rights reserved

ניהול מפתחות,
Public Key Infrastructure

מפתחות קריפטוגרפיים

בחיים ישנה ההנחה כי המפתחות הקריפטוגרפיים שלנו הם "טובים":

- נוצרו באופן אקראי טוב (מפתחות סימטריים הוגרלו באקראי באופן בלתי תלוי, מפתחות אסימטריים נוצרו בתהליכים דומים).
- משמשים לצרכים שנקבעו (מפתח MAC משמש ל-MAC ולא להצפנה בצופן בלוקים), כמו כן מומלץ להשתמש במפתחות שונים לכל צורך.
- אורכם גדול דיו כדי להיות מוגנים בפני סוגי התקפות שונים (לדוגמא, חיפוש ממצה).

כל הזכויות שמורות © All rights reserved

תכונות של מפתח קריפטוגרפי

לכל מפתח קריפטוגרפי מספר תכונות חשובות:

- אורך המפתח תלוי ב:
  - סוג האלגוריתם - צופן סימטרי, MAC, צופן פומבי, או אלגוריתם חתימה דיגטלית
  - סוג המידע המוצפן - ארוך יותר ככל שהמידע יותר סודי
  - זמן החיים של המידע – כמה זמן המידע צריך להשמר סודי/חתום

- אורך חיים
  - משך הזמן בו המפתח בשימוש או כמות המידע המוצפן באמצעותו
  - תלוי באורך החיים של המידע המוגן, והנזק שיגרם במקרה של חשיפה

כל הזכויות שמורות © All rights reserved

ניהול מפתחות קריפטוגרפיים

את המפתחות הקריפטוגרפיים יש לנהל בצורה בטוחה ואמינה:

- ייצור בטוח
  - מפתחות אקראיים לחלוטין
  - שימוש באמצעים פיסיים לייצור
- אחסון בטוח
  - שמירת המפתחות כך שלא יחשפו (רכיב נפרד)
  - עדיף שהרכיב הפיסי לייצור ישמש גם לשמירה
- הפצת מפתחות (נראה מיד)
  - החלפת מפתחות סימטריים בין אנשים המעוניינים לתקשר באופן בטוח
  - הפצת מפתחות אסימטריים (ציבוריים) לצורכי הצפנה אסימטרית וחתימות

All rights reserved © כל הזכויות שמורות

ניהול מפתחות קריפטוגרפיים

- עדכון מפתחות
  - תוקפם של מפתחות יכול לפוג, ואז יש צורך לעדכנם
  - המפתח הציבורי צריך להיות כזה שניתן לבצע לו Revocation – רישום כמפתח שאיננו חוקי יותר, אף על פי שהוא עדיין בתקופת הקיום החוקית שלו (לדוגמא, המפתח הפרטי נחשף בטעות)
  - זיהוי בעלי המפתח הציבורי (התקפות התחזות ופלגיאט)

כל הזכויות שמורות © All rights reserved

אלגוריתם הסכמה על מפתחות

נתונים שני אנשים A ו-B המעוניינים להסכים על מפתח סודי (סימטרי) תוך שימוש בערוץ פומבי בלבד.

- דרך אפשרית: הצפנת המפתח הסימטרי הסודי על ידי RSA ושליחתו בערוץ הפומבי (ניתן להצפין תחת כל פונקציית הצפנה פומבית בטוחה) – השיטה שמתבצעת ב-PGP.
- דרך נוספת: אלגוריתם יעודי להחלפת מפתחות – שיטת Diffie-Hellman

כל הזכויות שמורות © All rights reserved

אלגוריתם Diffie-Hellman להסכמה על מפתח סימטרי משותף

אלגוריתם Diffie-Hellman הוצע בשנת 1977 לצורך יצירת מפתח סודי משותף תוך שימוש בערוץ פומבי.

לכל אחד מהצדדים מפתח ציבורי ומפתח פרטי מתאים.

כל אחד מהצדדים מחשב את המפתח הסודי המשותף תוך שימוש במפתח הפרטי שלו, ובמפתח הציבורי של הצד השני.

כל הזכויות שמורות © All rights reserved

אלגוריתם Diffie-Hellman (המשך)

יהי p מספר ראשוני ויהי g יוצר של Z_p^*, החבורה הכפלית של Z_p. הפרמטרים g ו-p ידועים לכל משתמשי המערכת.

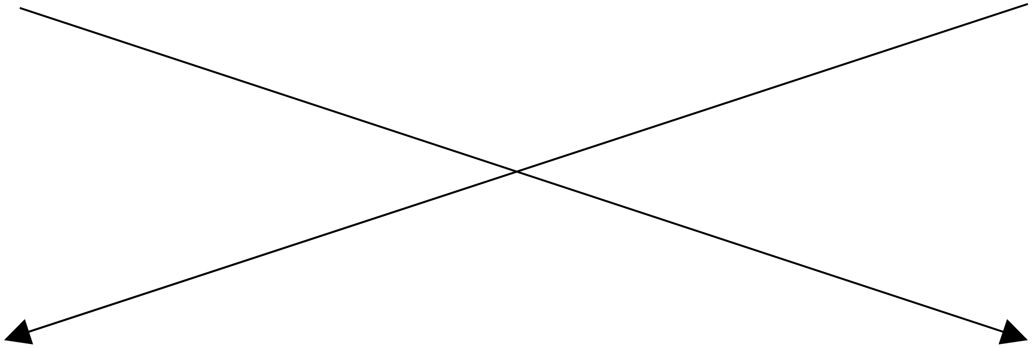
המפתחות:

	המפתח הפרטי	המפתח הפומבי
אליס	y	$g^y \pmod p$
בוב	x	$g^x \pmod p$

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

אלגוריתם Diffie-Hellman (המשך)

	אליס	בוב
(1	$y, g^y \pmod p$	$x, g^x \pmod p$
(2	$g^y \pmod p$	$g^x \pmod p$



(3	$g^{xy} \pmod p$	$g^{yx} \pmod p$

כל הזכויות שמורות © All rights reserved

אלגוריתם Diffie-Hellman - רקע

ההנחה המובלעת מאחורי אלגוריתם Diffie-Hellman היא שבהנתן g,p,g^x ו-g^y קשה לחשב מתוכם את g^{xy} (כל החישובים הם mod p).

בעיה זו (בעיית DH) קשורה לבעיית ה-DLOG (הלוגריתם הדיסקרטי): בהנתן g,p,g^x מצא את x.

כמו בעיית RSA (ובעיית הפירוק לגורמים) גם בעיות אלה נחשבות קשות, ועד היום אין אלגוריתמים טובים לפתרונן.

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

אלגוריתם Diffie-Hellman - אזהרות

שימו לב כי אלגוריתם Diffie-Hellman משמש אך ורק להסכמה על מפתח סימטרי משותף בין שני משתתפים.

האלגוריתם **איננו** יכול לשמש עבור:

1. הצפנת מפתח פומבי
2. חתימות דיגטליות
3. כל דבר שאיננו הסכמה על מפתחות

הערה: שימו לב כי שני אנשים המשתמשים תמיד באותם מפתחות פומביים (ופרטיים מתאימים) יסכימו תמיד על אותו מפתח משותף.

כל הזכויות שמורות © All rights reserved

Public Key Infrastructure (PKI)

- PKI הינה תשתית להעברה, פרסום ואימות של מפתחות ציבוריים.
- המפתחות הפומביים בדרך כלל מפורסמים בעזרת סרטיפיקטים.
- הגדרה של PKI צריכה להכיל הגדרות של התהליכים הבאים:
  - מתן סרטיפיקט (certification): מי מנפיק סרטיפיקט למי, תהליך ההנפקה, ואיפה הסרטיפיקט שמור.
  - אימות מפתח פומבי: כיצד מוצאים סרטיפיקטים של ישויות במערכת וכיצד מאמתים סרטיפיקט (האם הוא תקף).
  - ביטול הסרטיפיקט.

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

פרוטוקולי הזדהות קריפטוגרפיים

- ראינו קודם לכן, כי יש פעולות שרק קבוצה מסוימת של אנשים מסוגלת לבצע – לדוגמא, חתימה דיגטלית ניתנת לביצוע רק על ידי מחזיק המפתח הפרטי.
- ניתן לנצל תכונות אלה לצורך זיהוי – אם אנחנו מעוניינים לבדוק האם אדם מסוים הוא עצמו (דרך האינטרנט), נוכל לשלוח לו "אתגר" קריפטוגרפי שרק הוא ידע לפתור.
- רעיון זה נמצא בבסיסם של לא מעט פרוטוקולי בקרת כניסה – Kerberos, PPTP, SRP, EKE ואפילו בצורה מסוימת ב-SSH.

מועדון הלינוקס החיפאי
כל הזכויות שמורות © All rights reserved

הוכחות באפס מידע

- הרעיון הבסיס מאחורי הוכחות באפס מידע (Zero Knowledge Proof) הוא העובדה כי ניתן להוכיח טענה מסוימת ללא חשיפת ההוכחה עצמה.
- לדוגמא, כאשר מוכיחים טענה מתמטית מסוימת, אנחנו מצרפים הוכחה כך שכל אדם יוכל לבדוק את ההוכחה.
- בחלק מן המקרים, ניתן לתת הוכחה "הסתברותית", כזו שתשכנע אותנו בסיכוי נורא גבוה, אך לא תחשוף את דרך ההוכחה.
- לדוגמא, הקוסם הודיני טען שהוא מסוגל להשתחרר מכל קשר שהוא. במשך הרבה מאוד פעמים הוא הוכיח יכולת זו, בלי לחשוף את צורת פעולתו.

כל הזכויות שמורות © All rights reserved

הוכחות באפס מידע (המשך)

- אפשר להגדיר את מחלקת השפות שניתנות להוכחה באפס מידע. מחלקה זו כוללת בתוכה את NP, ולכן ניתן להוכיח כי אנחנו יודעים אלגוריתם הפותר את SAT, גם ללא חשיפתו.
- מקרה פרטי של הוכחות באפס מידע, הוא הוכחות מקרה באפס מידע (zero knowledge arguments). במקרה זה, אנחנו מוכיחים ידיעת פתרון אחד מסוים לבעיה, מבלי לחשוף את אותו פתרון.
- לדוגמא, בעיית ה-3-צביעה של גרף היא בעייה NP-שלמה. בהנתן גרף ושלוש צביעה שלו, ניתן להוכיח את הידיעה הזו, ללא חשיפת ה-3-צביעה עצמה.

כל הזכויות שמורות © All rights reserved

מקורות נוספים

- אתר הקורס קריפטולוגיה מודרנית (236506) - http://www.cs.technion.ac.il/~cs236506
- הספר – Cryptography: Theory and Practice (מהדורה ראשונה!) מאת D.R. Stinson.
- הספר – The Code Breaker מאת David Kahn.
- אתר International Association of Cryptlogic Research http://www.iacr.org.

- לא להציץ ב-sci.crypt! איכות הדיון והמידע המפורסמים שם היא נמוכה עד שלילית.
- הספר של ברוס שנייר (Applied Cryptography) הוא בעייתי עקב ריבוי שגיאות. עם זאת אם מתעלמים מכל האספקטיים הטכניים, הוא סביר למדי.

כל הזכויות שמורות © All rights reserved