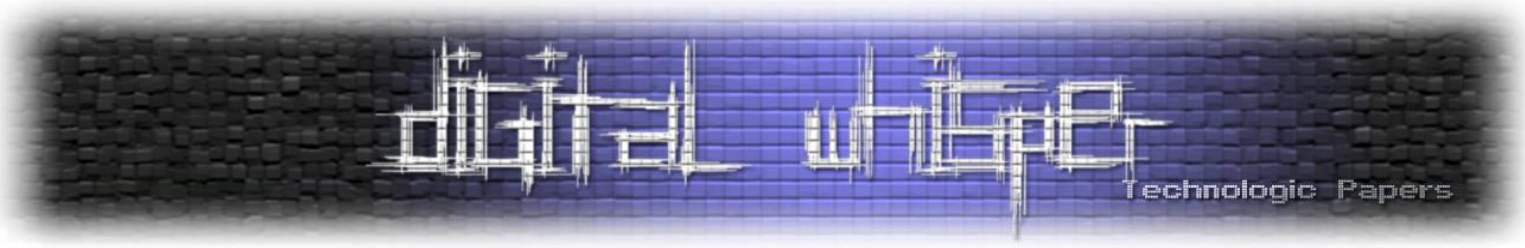


Hash Collisions

מאת גדי אלכסנדרוביץ' ואפיק קסטיאל (cp77fk4r)

הקדמה

דרך פשוטה ביותר לבזוז כסף מההמון באמצעות האינטרנט היא התחזות.

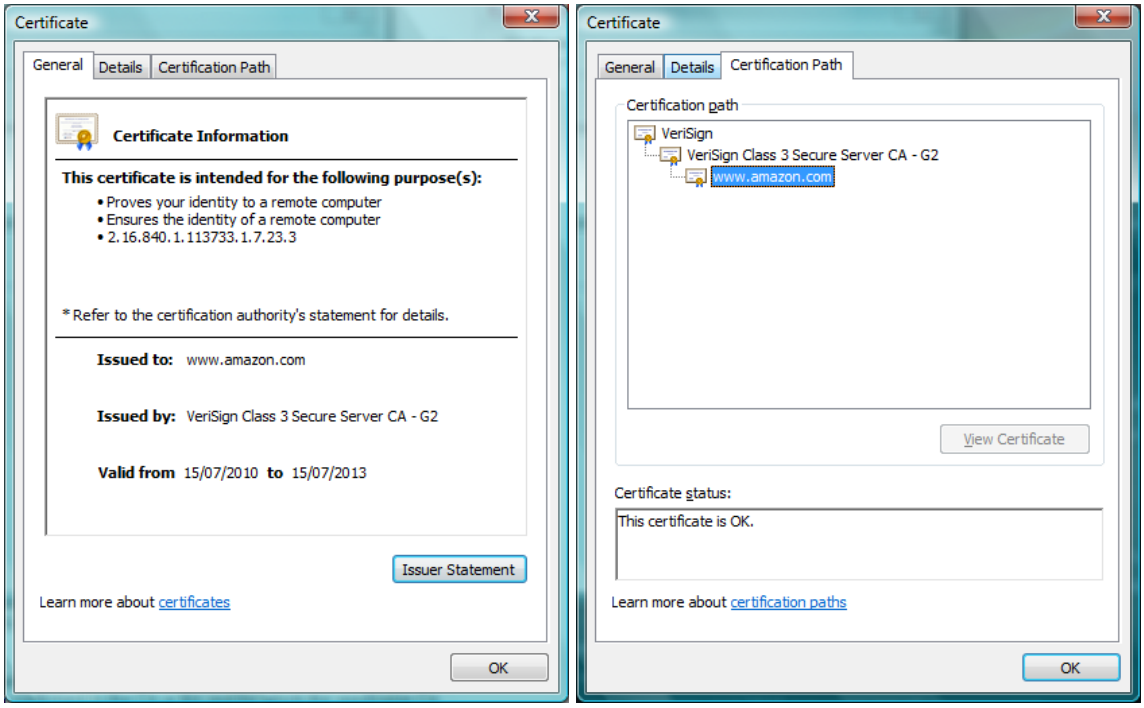
אני יכול לבנות אתר שנראה כמו עותק מדוייק (פחות או יותר...) של, נאמר, אמאזון, ולגרום ללקוחות מסכנים להיכנס אלי בטעות (נניח, על ידי בחירת שם מתחם דומה לזה של אמזון, או אפילו על ידי התקפות אלימות יותר כמו "הרעלת" שרתי DNS - השרתים שבהם משתמשים על מנת לתרגם כתובת מילולית לכתובת המספרית שאליה מתחברים בפועל). הלקוח התמים יבצע את העסקה וישלח את פרטי כרטיס האשראי שלו מבלי להבין שמשהו השתבש. אני אוכל אפילו לבצע עבורו את הקניה באופן חוקי לגמרי מאמזון עצמה, בעזרת המידע שהוא נתן לי, ובכך לוודא שהוא לא ישים לב כלל לרמאות - אבל אני אוכל לחגוג על חשבון כרטיס האשראי שלו אם וכאשר ארצה. השם המקובל להתקפה כזו בעברית יפה הוא "פישינג" ("דיוג"בלעז).

לפי דו"ח איומי האינטרנט לשנת 2010 (חציון ב') שפורסם בשיתוף עם החברות Alt-n Technologics ו-Commtouch, נשלחים בממוצע כמעט 180 מיליארד אימיילים הנושאים עמם תוכן ספאם או מתקפות דיוג ביום.

איך אפשר למנוע את זה?

מה שאנחנו רוצים הוא שיטת אימות כלשהי - דרך לוודא שגורם מסויים הוא מי שהוא מתיימר להיות. דרך פשוטה לוודא זאת היא זו: לנהל את התקשורת הסודית (זו שכוללת פרטי כרטיס אשראי וכדומה) עם האתר באופן מוצפן, כשההצפנה היא באמצעות מפתח סודי שידוע רק לאתר עצמו - במילים אחרות, אנחנו יודעים כיצד להצפין הודעות אך רק האתר יודע כיצד לפתוח אותן. שיטת הצפנה שכזו מכונה "הצפנת מפתח פומבי" (כי המפתח - המידע שאנו משתמשים בו כדי לבצע את הנעילה - הוא פומבי וידוע לכל).

לרוע המזל, יש כאן בעיה של ביצה ותרנגולת - איך נדע מהו המפתח הפומבי של אמזון? תגידו, הם יאמרו לנו מהו - אבל אם אנחנו מתעסקים עם אתר מתחזה, איך נוכל להיות בטוחים שהמפתח הוא אכן המפתח שלהם ולא של המתחזה? זו בעיה לא פשוטה כלל, ואין לה פתרונות אלגנטיים - הפתרון המקובל כיום באינטרנט הוא שגורמים מוסמכים ינפיקו אישורים (Certificates) - כל אישור כולל את שם האתר, את המפתח הפומבי שלו ועוד אי אלו פרטים מזהים, ובנוסף לכך חתימה של הגורם המוסמך על האישור. אם יתברר שהייתה תרמית, הגורם המוסמך יישא עליו את האחריות.



(החתימה הדיגיטלית של Amazon שהוחתמה ע"י VeriSing)

האופן שבו ניתן לבצע "חתימות" דיגיטליות שכאלו ראוי למאמר בפני עצמו ולא ארחיב עליו, אך הרעיון הבסיסי הוא שבהינתן מידע כלשהו ניתן לבצע עליו אי אלו מניפולציות ולקבל פלט שהוא ה-"חתימה" המתאימה, כך שניתן לוודא באופן מסויים שהחתימה היא אך ורק של מי שהתיימר לחתום (בשביל לעשות זאת צריך לדעת מהו המפתח הפומבי שלו, אך עבור מנפיקי אישורים זה כבר מידע שנמצא בכל דפדפן, פחות או יותר) ובאופן כזה שאם משנים אפילו ביט אחד ויחיד מהמידע שעליו חותמים, החתימה הופכת מייד לחסרת ערך.

אלא שבדרך כלל לא חותמים על המידע עצמו, אלא על ה**תמצית** שלו, ועל זה אני רוצה לדבר הפעם. מהי המשמעות של תמצית, למה צריכים אותה, והחשוב מכל - **איך אפשר לתקוף אותה ספציפית**?

נתחיל ממה זה. תמצות הוא פעולה שבה לוקחים קובץ מידע כלשהו (שיכול להיות קובץ טקסט בן מספר שורות, קובץ וידאו בן מאות מגה בייטים, או קובץ התקנה של ווינדוס בן כמה ג'יגות) ומפעילים עליו פונקציה שמחזירה מחרוזת ביטים קצרה (ביט הוא או 0 או 1), נניח באורך 256 ביטים. על המחרוזת הזו ניתן לחשוב כעל מעין "חתימה ייחודית" של הקובץ, מה שנשמע מאוד מאוד מוזר בהתחלה - כיצד יכולה מחרוזת ביטים קצרצרה לתאר במדוייק קובץ? התשובה היא שאכן היא לא יכולה, אבל זה לא מפריע לנו.

חשבו על טביעת אצבע - למרות שטביעת אצבע מכילה מעט מאוד מידע על האדם שלו היא שייכת, היא מזהה אותו כמעט במדויק. אפשר תמיד לשאול את השאלה מה מבטיח לנו שלא יהיו שני אנשים בעלי אותה טביעת אצבע, והתשובה היא שייתכן שיהיו אך זה מאוד לא סביר.

קצת מתמטיקה

כדי להבין למה זה תקף גם לתמציות צריך לעשות קצת חשבון. הבה ונניח שכל התמציות שלנו הן אכן בגודל 256 ביטים, וכמו כן נניח שפונקצית התמצות פועלת באופן כזה שערך התמצית שכל קובץ מקבל נראה אקראי לחלוטין - כאילו הגרלנו אחת מהמחרוזות מאורך 256 הקיימות. כמה תמציות ישנן? חשבון פשוט מראה ש-2^{256}. המספר הזה נראה מרשים יותר כשכותבים אותו במפורש:

$$115792089237316195423570985008687907853269984665640564039457584007913129639936$$

האנלוגיות הרגילות עובדות פה - זה מספר שדי קרוב בגודלו למספר האטומים המוערך ביקום. עכשיו, כמה קבצי מחשב כבר נוצרו, במהלך כל ההיסטוריה של המין האנושי? מיליארד מיליארד מיליארדים? מיליארד פעמים המספר הזה? מיליארד פעמים המספר הזה? זה עדיין לא מגרד אפילו את 2^{256}. בקיצור - יש מקום לכולם. אם הייתה למעלה יד מכוונת היא לא הייתה מתקשה לתת מזהה ייחודי בן 256 ביטים לכל קובץ שאי פעם נוצר. הבעיה היא שאנחנו לא מכירים יד שכזו ולכן צריכים להשתמש בפונקציות מעשה ידי אדם - פונקציות שצריכות להיות פשוטות ומהירות ככל הניתן.

אז זה מה שזה. אבל למה צריך את זה? כפי שכבר נאמר, אנחנו רוצים להתאים סימן מזהה לכל פיסת מידע בעולם. שימוש אחד לדוגמה של המידע הזה הוא בהורדת קבצים - סיימנו להוריד קובץ בן ארבעה ג'יגה ואנחנו רוצים לוודא שהכל בסדר איתו? אנחנו מחשבים את התמצות שלו ומשווים לתמצות שנשמר באותו אתר שסיפק לנו את הקובץ (יש לפעמים אתרים שעושים את זה. באמת!). אם הם שווים, טוב. אם לא - כנראה שמשהו השתבש (בזדון או בטעות) בקובץ שהורדנו. גם תוכנות כיווץ נוהגות להשתמש בסוגים פשוטים של תמצות כדי לבדוק שהקבצים שנפתחו אכן מתאימים למה שכווץ ולא ארעה תקלה כלשהי - אם יצא לכם להוריד קובץ מכווץ ולהיתקל בהודעת שגיאה מוזרה שהכילה צירוף מילים כמו CRC בתוכו - התוכנה התלוננה על כך שהתמצות השמור בקובץ לא מתאים למידע שחולץ ממנו.

שימוש אחר הוא בהקשר שכבר תיארתי של חתימות. הבעיה בחתימות היא ששיטות החתימה הקיימות פועלות לרוב על פיסות מידע קטנות מאוד - לא יותר ממאות ביטים. כדי להשתמש בחתימה על פיסת מידע גדולה יותר צריך לפרק אותה לפיסות קטנות ואז לפעול על כל פיסה בנפרד. זו גישה בעייתית מאוד, כי פירושה הוא שאפשר יהיה לשחק משחקי "העתק-הדבק" עם החתימה. למשל, נניח שיש לנו אישור של מפתח פומבי שמכיל את השם והכתובת של האתר שלנו, ואת המפתח הפומבי שלנו, והחתימה היא על שתי פיסות המידע הללו בנפרד - אנחנו מקבלים חתימה אחת על השם והכתובת, וחתימה אחרת על המפתח. מה שנעשה, אם אנחנו רוצים לרמות, יהיה לקחת את האישור של אמזון - שגם הוא מכיל חתימה נפרדת על השם והכתובת, וחתימה נפרדת על המפתח, וליצור אישור "משולב" שמכיל את השם והכתובת של אמזון, ואת המפתח הפומבי שלנו... התוצאה? אישור חתום למהדרין שהוא חלומו של כל האקר. בקיצור, זה לא עסק. אי אפשר לפרק את המידע לפיסות ולחתום על כל אחת בנפרד - צריך איכשהו "לערבב". הפתרון המתבקש ביותר הוא לקחת את כל המידע, להפעיל עליו תמצות, ואז לחתום על התמצית עצמה - שהיא כל כך קטנה שאין צורך לפרק אותה לכלום.

דוגמה טובה לכך ניתן לראות בחולשה שנמצאה לא מזמן במנגנון ההחתמה המלווה בפורמט הקבצים PDF (מגרסה 1.3). הדוגמה התגלתה על ידי חוקר האבטחה הגרמני Florian Zumbiehl ופורסמה בעשירי לאוגוסט ב-bugtraq, ניתן לקרוא אודותיה בעמוד הבא:

http://pdfsig-collision.florz.de/

בדוגמה ניתן לקרוא כי אותו חוקר הצליח ליצור שתי הודעות בפורמט PDF המכילות מידע שונה בתכלית, המשתפות חתימה דיגילית אחת:

Julius. Caesar Via Appia 1 Rome, The Roman Empire May, 22, 2005 To Whom it May Concern: Alice Falbala fulfilled all the requirements of the Roman Empire intern position. She was excellent at translating roman into her gaul native language, learned very rapidly, and worked with considerable independence and confidence. Her basic work habits such as punctuality, interpersonal deportment, communication skills, and completing assigned and self-determined goals were all excellent. I recommend Alice for challenging positions in which creativity, reliability, and language skills are required. I highly recommend hiring her. If you'd like to discuss her attributes in more detail, please don't hesitate to contact me. Sincerely, Julius Caesar	Julius. Caesar Via Appia 1 Rome, The Roman Empire May, 22, 2005 Order: Alice Falbala is given full access to all confidential and secret information about GAUL. Sincerely, Julius Caesar

למי שמעוניין לבדוק זאת, ניתן להוריד את הקבצים מכאן:
http://pdfsig-collision.florz.de/letter_of_rec.pdf
http://pdfsig-collision.florz.de/order.pdf

ומכאן ניתן להוריד קישור ל-Root CA (לא לשכוח להסיר אותו לאחר מכן):
http://pdfsig-collision.florz/rootca.pem

כמו שניתן לראות, על ידי ניצול החולשה ניתן לזייף את תוכן ההודעה וכל זאת מבלי לפגוע בפלט שנקבל בפעולת תמצות. כך נוכל לשנות תוכן/מלל של קובץ מסויים, ובכל זאת לא נתגלה על ידי מנגנוני הזיהוי הקיימים בו.

דוגמה נוספת לכך ניתן לראות בחשיפה שגילה לאחרונה חוקר האבטחה HD Moore במנגנון ההזדהות של מערכת ההפעלה VxWorks (מערכת embedded). בתחילת דרכה, המערכת הייתה מאחסנת את סיסמאות המשתמשים כ-Clear-Text, כך כל משתמש בעל גישה לאיזור אחסון הסיסמאות היה יכול בקלות לדלות את סיסמאותיהם של שאר המשתמשים במערכת. בכדי למנוע זאת, Wind River מימשו מנגנון בשם vxencrypt (ניתן לראות את הקוד כאן) לגיבוב הסיסמאות, כך שהמערכת הייתה מאחסנת תמצות של הסיסמאות במקום את הסיסמאות עצמן.

פעולת ההזדהות למערכת דרך ממשק Telnet או חיבור FTP מתבצע באופן כזה שכאשר המשתמש מכניס את סיסמתו בממשק ההזדהות, הסיסמה עוברת גיבוב על-פי אותה הפונקציה שבה המערכת השתמשה בכדי לאחסן את הסיסמה המקורית ורק לאחר מכן מתבצעת השוואה בין תמצות הקלט שהמשתמש הכניס לבין המחרוזת המאוכסנת במערכת. במידה וההשוואה הייתה חוזרת חיובית-המערכת מאפשרת למשתמש להתחבר למערכת.

כך, גם ניתן לבצע הזדהות למערכת באופן מאובטח וגם משתמש בעל גישה לאיזור אחסון הסיסמאות במערכת אינו יכול לדעת מה הן הסיסמאות של שאר המשתמשים.

עד כאן הכל טוב ויפה, אך החולשה ש-HD Moore מציג, מראה כי אותה פונקציית גיבוב סובלת מאחוז גדול מאוד של התנגשויות, זאת אומרת שכאשר משתמשים בפונקציה זו, ניתן לקבל את אותו הפלט (מחרוזת מגובבת) ממספר קלטים שונים (סיסמאות שהמשתמש מכניס). HD Moore הצליח לזהות את החולשה הזאת כפוטנציאל למימוש Brute Force.

מדיניות קביעת הסיסמאות במערכת ההפעלה דורשת מהמשתמש להכניס סיסמה בת לפחות 8 תווים ולא יותר מ-40 תווים, כך שלמעשה כמעט בלתי ניתן לנחש את סיסמאות המשתמשים. HD Moore הציג כי למעשה ישנן "בסביבות ה-210,000" סוגי פלטים שונים לפונקציה, כך שאין צורך לנחש את הסיסמה המקורית, אלא מספיק לנחש את אחד מ-210,000 הקלטים שיגרום להתנגשות וכך לגרום לפונקצית ההשוואה להחזיר תשובה חיובית בתהליך ההזדהות למערכת.

בדוגמה ש-HD Moore הציג ב-Bugtraq ניתן לראות כי כאשר המחרוזת "insecure" נשמרת כסיסמה במערכת ההפעלה, היא נשמרת כ-"Ry99dzRcy", כהמשך לאותה הדוגמה, HD Moore הציג כי גם פעולת תמצות על המחרוזת "s{{{{{^O" תשיג תוצאה זהה.

בהתחשב בעובדה שהמערכת המדוברת אינה תומכת בנעילת חשבונות אשר זוהו כמותקפים, ניתן לפצח כל חשבון בלא יותר מחצי שעה.

כעת נשאלת השאלה- מהי פונקצית תמצות "טובה"? בפרט, אם אנחנו רוצים להשתמש בפונקצית התמצות בשימושים קריפטוגרפיים, ולכן צריכים להביא בחשבון שבצד השני יש האקר מתוחכם מאוד, מה אנחנו צריכים שהפונקציה תקיים? ראשית, היא חייבת להיות קלה לחישוב, אבל פרט לכך לרוב מונים שלוש תכונות עיקריות: שיהיה קשה להפוך אותה, שיהיה קשה למצוא לה התנגשויות, ושיהיה קשה למצוא עבורה תמונה שנייה (Second Preimage). בואו נעבור על כל תכונה בנפרד.

"להפוך" פונקציה פירושו למצוא קלט אפשרי בהינתן פלט נתון. למשל, הייתי רוצה למצוא איכשהו קלט לפונקצית התמצות שלי שיוציא את הפלט 0. פונקצית תמצות טובה תהיה כזו שגם אם אני יודע בדיוק איך היא עובדת, זה עדיין קשה לי מאוד למצוא קלט כזה - בתקווה, לא יהיה לי שום דבר חכם יותר לעשותו מלבד להזין לפונקציה עוד ועוד ערכים ולהתפלל שיצא הפלט 0 (ואם אכן הפלטים מתפלגים באופן שנראה אקראי, זה ייקח לי זמן רב - בסביבות ה-2^{128} נסיונות לפני שאצליח - הרבה, הרבה יותר ממספר השניות שחלפו מאז ראשית היקום). למה זה מסוכן? ובכן, נניח שזיהיתי חולשה כלשהי בשיטת החתימה שאני רוצה לתקוף, אבל כדי לנצל אותה החולשה הזו דורשת שהחתימה תהיה על תמציות עם תכונות מאוד, מאוד ספציפיות (כה ספציפיות עד שבפועל הסיכוי שהן יופיעו כשחותמים על משהו לגיטימי הוא אפסי). אם אוכל להפוך את פונקצית התמצות, אוכל לייצר קלט כזה שהפלט של פונקצית התמצות עליו (ולכן, מה שנחתם בפועל) הוא בדיוק מה שמאפשר לי לייצר זיוף.

כמובן שאפשר לשאול איך בדיוק הקלט הזה ייראה - הרי ככל הנראה הוא ייראה כמו ג'בריש מוחלט ולא כמו הודעת טקסט יפה. אז מה הבעיה? הבעיה היא שלפעמים מצפים למצוא חתימה על ג'בריש מוחלט. כאשר חותמים על מפתחות הצפנה פומביים, מצפים שמפתח ההצפנה יהיה ג'בריש. ויש עוד סיטואציות שבהן ג'בריש לא נראה מופרך. כמובן, חלק מההתמודדות עם הבעיה הזו היא שבפרוטוקולים שלנו לא יהיה מקום להודעות שהן ג'בריש מוחלט ותמיד חייב להיות רכיב לא-ג'בריש בהן, אבל כאן אנחנו כבר גולשים לדיון על מתקפת הנגד למתקפת הנגד.

"למצוא התנגשות" פירושו לייצר יש מאין שני קלטים שמתומצתים לאותו הפלט - כאשר כאן לא חשוב הפלט (כלומר, אנחנו לא מנסים להפוך את הפונקציה) אלא רק שהוא זהה עבור שני הקלטים. דרך פשוטה אחת לעשות זאת היא לייצר $2^{256}+1$ קלטים אקראיים ואז מובטח לי שעבור פונקצית התמצות שלי (שבה תמצית היא מגודל 2^{256} ביטים) תהיה התנגשות אחת לפחות - זהו "עקרון שובך היונים" המתמטי. רק מה, 2^{256} זה הרבה, כך שהשיטה הזו לא שווה הרבה בפני עצמה. אלא שמתברר, באופן אולי מפתיע, שלא צריך לייצר עד כדי כך הרבה קלטים כדי שיהיה סיכוי סביר להתנגשות, ואתאר זאת באמצעות האנלוגיה המקובלת - ימי הולדת.

אפשר לחשוב על ימי הולדת כעל פונקצית תמצות (גרועה) - לכל אדם מתאימים מספר שהוא תאריך יום ההולדת השנתי שלו. אם מניחים שאנשים לא נולדו ב-29 (או ב-30) בפברואר, יש לנו 365 אפשרויות. מה שאמרתי קודם הוא שאם יש לנו קבוצה של 366 אנשים, מובטח לנו שלפחות לשניים מהם יש אותו יום הולדת - אבל זו כמות גדולה יחסית של אנשים; נשאלת השאלה מה קורה אם אנחנו מוותרים על הדרישה שבודאות יהיו לנו שני אנשים בעלי אותו יום הולדת ומסתפקים בדרישה שבהסתברות לא רעה יהיו שניים עם אותו יום הולדת - נניח, הסתברות של חצי? אם מניחים שימי ההולדת מתפלגים באופן אחיד (הנחה שאינה נכונה במציאות, אבל מילא) מתברר שצריך הרבה, הרבה פחות אנשים - רק 23. במילים אחרות - בכל כיתה בבית ספר יש סיכוי גדול מחצי שיש שני תלמידים עם אותו יום הולדת, כלומר אם ניקח בית ספר ונשאל את עצמנו "בכמה מהכיתות יש שני תלמידים עם אותו יום הולדת?" ככל הנראה נגלה שבערך בחצי מהכיתות זה קורה (כמובן שאם תעשו את הניסוי הזה והוא ייכשל אני אוכל להתחמק ולטעון שיש את ה-29 בפברואר, וימי הולדת לא מתפלגים אחיד, ובכלל שמסוכן מאוד לקשר בין מתמטיקה והעולם האמיתי).

פורמלית התוצאה הזו נקראת "פרדוקס יום ההולדת" והיא אומרת שאם יש לנו תמצית עם N תוצאות אפשריות, אז מספיק שיהיו לנו בערך $\sqrt{N}$ קלטים כדי שתהיה הסתברות טובה להתנגשות. לכן כדי למצוא התנגשות לפונקצית התמצות שתיארתי קודם מספיק לי לייצר רק 2^{128} איברים. גם זה מספר אדיר למדי, אבל זה לא מקרי: פרדוקס יום ההולדת הוא אחד מהשיקולים בתכנון פונקציות תמצות, ובפרט בכך שבוחרים את מרחב התמצות להיות גדול למדי (ביחס, נניח, למרחב המפתחות שבו משתמשים בהצפנות סימטריות כמו AES).

את הסכנה שבמציאת התנגשויות קל לתאר באמצעות דוגמה: נניח שאני מצליח לייצר שתי הודעות, האחת אומרת "אני חייב לך 10 ש"ח" והשניה אומרת "אני חייב לך 1,000,000 ש"ח" שלשתיהן אותה תמצית ואני גורם לכם לחתום על ההודעה הראשונה - בכך יצרתי חתימה (חוקית לחלוטין) שלכם גם על ההודעה השניה! כמובן שזו רק המחשה נאיבית; יש דרכים מתוחכמות בהרבה לנצל שני קלטים שמתומצתים לאותו הערך.

"מציאת תמונה שנייה" נראית במבט ראשון זהה רעיונית גם להיפוך וגם למציאת התנגשות. פירושה שבהינתן קלט x מסויים, נוכל למצוא עבורו קלט y שמתומצת לאותו דבר. זה לא בדיוק היפוך או מציאת התנגשות, שכן אין לנו שליטה על x - הוא קבוע ונתון, ואנחנו יכולים לשחק רק עם y. זו ההתקפה שהכי מזכירה את מה שדיברתי עליו בהתחלה - במקרה הזה x הוא האישור החוקי של אמזון, ואני רוצה לייצר אישור מזוייף עבור עצמי, y, שהתמצית שלו תהיה זהה לזו של x, ולכן החתימה הקיימת על x תהיה חתימה חוקית גם עבור y שלי.

אם כן, אלו הדרישות שלנו מפונקצית תמצות. האם קיימות בכלל פונקציות שעומדות בדרישות הללו? התשובה היא "כן, לא, ואנחנו לא יודעים". כן, בגלל שהפונקציות המתקדמות ביותר שבהן משתמשים כיום עדיין לא נפרצו - עדיין לא הוצגה התקפה שבה נשברת אחת מתכונות אלו; לא, בגלל שפונקציות תמצות רבות כן נשברו - התכונה שאותה מצליחים לשבור היא כמעט תמיד מציאת התנגשויות, ופונקציות תמצות רבות שעדיין משתמשים בהן כיום הן כבר לחלוטין לא בטוחות מהבחינה הזו. הדוגמה הבולטת ביותר היא הפונקציה MD5 שנמצאת בשימוש נרחב מאוד אך כבר הודגם כי באופן פרקטי ניתן לנצל את החולשות שלה כדי לזייף אישורים. בקיצור - בחירת פונקצית התמצות עדכנית ובטוחה היא מאוד חשובה ולא ניתן סתם להסתמך על מה שכולם משתמשים בו כיום.

ה-"אנחנו לא יודעים" מתייחס לפער הגדול שיש בין התיאוריה לפרקטיקה בתחום הקריפטוגרפיה. כאשר עוברים לתיאוריה, יש הגדרות מתמטיות מדוייקות למושגים האמורפיים של "קשה להפוך פלט אקראי" (מה זה קשה? איך מודדים אותו? איך משפיעה האקראיות של הפלט), ותחת ההגדרות המחמירות הללו עדיין לא ברור אם בכלל קיימת פונקציה שעונה עליהן או לא. למעשה, פשוט למדי להראות שאם תהיה הוכחה מתמטית לקיום של פונקצית תמצות טובה (או אף פחות מכך - סתם פונקציה שמקיימת את תכונת קושי ההיפוך - מה שנקרא "פונקציה חד-כיוונית") הדבר יפתור מייד את השאלה הפתוחה המרכזית במדעי המחשב - השאלה האם $\mathrm{P}=\mathrm{NP}$ (שאלה שראויה לסיקור נפרד) - והתשובה תהיה "לא" רועם (שכן אם $\mathrm{P}=\mathrm{NP}$ אז ניתן להפוך בקלות כל פונקציה שקל לחשב).

אם כן, פונקציות תמצות מעניינות גם מההיבט המעשי וגם מההיבט התיאורטי ובכל הקשור אליהן - כמו ברוב הנושאים במדעי המחשב - עדיין רב הנסתר על הגלוי.