מבוא לקריפטוגרפיה - תרגיל בית 3

נא להגיש עד ה 3.3.2019 דרך המודל (כרגיל, בזוגות).

נושאים: קודים לאותנטיקציה של הודעות (MAC), מימוש מערכות הצפנה

1. בהרצאות האחרונות הגדרנו קודים לאותנטיקציה של מידע. (Message Authentication Codes). בניגוד למערכות הצפנה, מטרתם של קודים כאלה לוודא את נכונות המידע העובר בערוץ, במידה והתוקף מנסה לשנות אותו. נתנו הגדרה לקוד MAC עם בטיחות חישובית (זו תזכורת, ולא הגדרה פורמלית לחלוטין. ההגדרה בכיתה נוסחה באמצעות תוקף ו chellanger):

- Correctness: For all n and all $k \in K_n, m \in M_n$, $Pr[Verify(m, MAC(k,m)) = 1] = 1$.
- Existential unforgeability (short reminder): For any PPT adversary $A(1^n)$ obtaining an oracle access to $MAC_k()$, where k is picked at random from $Gen(1^n)$, manages to generate a pair m, t such that $Verify(k, m, t) = 1$ and m was not sent to the oracle as a query is negligible in n. The probability is taken over the random choice of k, and the randomness of the MAC_k oracle.

א. נתבונן במשפחה של קודי אותניקציה $MAC = (Gen_n, MAC_n, Verify_n)$ המוגדרת מעל הקבוצות $M_n = K_n = \{0,1\}^n, T_n = \{0,1\}^{10log(n)}$ המקיימת נכונות.

הוכיחו שמשפחה זו בהכרח אינה מקיימת existential unforgeability.

ב. בהרצאה בנינו MAC להודעות חסומות $M_n = K_n = T_n = \{0,1\}^n$. התחלנו ממשפחה של פונקציות פסוודו-אקראיות $f_n(k,m) : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. כזכור, $Gen(1^n)$ מייצר ייצוג k עבור פונציה אקראית במשפחה (במאצעות קביעת האינדקס k) $f_{n,k}(m)$, ו $MAC_k(m) = f_{n,k}(m)$. הוכיחו שהבניה אכן מקיימת נכונות ו existential unforgability.

2. תרגיל מימוש של RSA:

א. ממשו את RSA כפי שמתואר במשימה הבאה. http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Crypto/Crypto_RSA/Crypto_RSA.pdf יש לפתור את 3.1 עד 3.3 כולל. כלומר, לממש את Gen, Enc, Dec של "Textbook RSA" ללא רנדומיזציה (שכמובן לא בטוח כמו שהוא, אבל זה מספיק לצורך התרגיל). בשלב זה, תוכלו לממש Gen "דמה" בו המפתח קבוע למכפלה של ראשוניים גדולים כלשהם (למשל ממשימה 3.1, או מהאנטרנט).

ב. ממשו אלגוריתם למציאת מספרים ראשוניים גדולים. האלגוריתם יקבל את מספר הביטים במספר,יבחר ראשוני באורך זה באקראי (ללא אפסים מובילים), יבדוק ראשוניות, וחמזיר את הראשוני הראשון שימצא. את בדיקת הראשוניות יש לבצע באמצעות אלגוריתם Solovay-Strassen. מימוש שלו ניתן למצוא ב Wiki ובמסמך https://kconrad.math.uconn.edu/blurbs/ugradnumthy/solovaystrassen.pdf מומלץ לקרא את המסמך ולהבין מדוע Fermat test "פשוט" לא עובד (Carmichael primes), ולמי שמתעניין, לקרא ולהבין כיצד Solovay-Strassen פותר את הבעיה. יש לבנות אלגוריתם הטועה בהסתברות 0.001 לכל היותר (כלומר, מזה חסם על הסיכוי שיחזיר מספר שאינו ראשוני). לשם כך, יש להשתמש בחסם הטעות של Solovay-Strassen הבסיסי המופיע במאמר לעיל (נוסחא 2.1).

השתמשו בו לבניית Gen בסעיף 1.