המחלקה להנדסת חשמל

שם הפרויקט :

הטמעה של הגנות חומרה מבוססות רנדומיזציה בזמן במערכת קריפטוגרפית לצורך שיבוש זליגת המידע ומניעה התקפות ערוץ צד.

Project name :

Implementation of countermeasures based on time randomization into cryptographic hardware to prevent information leakage and to prevent side channel attacks .

ספר הפרויקט

שם הסטודנטים : עוז יוסף , עומר אביב

שם המנחה : ד"ר ויצמן יואב

חתימת המנחה : מצורף אישור מנחה

תאריך הגשה : 11/9/2021

תודות

בראש ובראשונה נרצה להודות למנחה שלנו , ד"ר יואב וימן על ההנחיה , הליווי ההכוונה , העזרה ,התמיכה והזמינות לכל שאלה שעלתה במהלך העבודה על הפרויקט ותרם רבות והצלחתו.

ברצוננו להודות ליהודה ולצוות המעבדה של בר אילן על השימוש וההסבר ברכיבי המעבדה .

ברצוננו להודות למשפחותינו אשר היו לנו לכתף תומכת במהלך ביצוע הפרויקט במהלך היום ואף בשעות המאוחרות של הלילה.

תודה רבה.

אישור הגשה :

RE: בדיקת ספר הפרויקט פוסטר וקיטלוג

YW **Yoav Weizman** <yoavw@afeka.ac.il>
6:10 PM

To: Oz Yosef; Yoav Weizman; Yoav Weizman; Omer Aviv

מאשר להגשה

בהצלחה ושנה טובה

תוכן עניינים

רשימות

איורים

טבלאות

נוסחאות

גרפים

מילון מונחים

מפתח הצפנה סודי – רצף תווים אשר משמש כבסיס ייחודי לתוצאת ההצפנה והינו נסתר.

AES - Advanced Encryption Standard, צופן בלוקים סימטרי.

FPGA (Field-Programmable Gate Array) - סוג של מעגל משולב אשר ניתן לשינוי.

Vivado – סביבת העבודה לכרטיסי FPGA של חברת Xilinx.

LFSR (Linear Feedback Shift Register) – רגיסטר שבו יחס ההתרחשויות ההולכות ונשנות במוצא תלוי במצבו הקודם.

RandomClock – שיטת יישום המאפשרת שימוש סלקטיבי במס שעונים באופן פסאודו אקראי בעזרת LFSR ומMuxBufferGlobal.

PLL (Phase Locked Loop)– רכיב מחולל שעונים כתלות בשעון הייחוס בכניסתו .

היפותזה – השערה שטרם ניתן לה אישור מספק.

מודלי כוח - Power models, מדדים כמותיים למידת שינוי של אות או סיגנל.

קורלציה – מתאם, תכונה של קשר סטטיסטי בין משתנים.

Cipher text – המידע המוצפן במוצא אלגוריתם ההצפנה.

Plain text – המידע המקורי הלא מוצפן אשר נכנס לאלגוריתם ההצפנה..

התקפות ערוצי הצד - התקפות קרטוגרפיות המנצלות מידע שמושג מאופן היישום הפיזי או השימוש של מערכת ההצפנה .

CPA (Correlation Power Analysis) - תקיפה המתבסת על עיקרון מציאת קורלציית בין רמות בPower.

DPA (Differential power analysis) - תקיפה המתבסת על ניתוח סטטיסטי של נתונים שנאספו מפעולות קרטוגרפיות מרובות.

CR – היחס בין הקורלציה של ההיפותזה הנכונה לבין הקורלציה השגויה הראשונה בגודלה.

S-Box – תיבת שיחלוף המהווה חלק מתהליך ההצפנה באלגוריתם AES.

Test bench – סביבת ריצה המיועדת לבדיקת תקינות אלגוריתם או מודל.

Nb – פונקציה של גודל הבלוק .

Nk - פונקציה של אורך המפתח .

Nr – מספר הסבבים הדרוש התלוי בערכי Nk,Nb .

תקציר:

מטרתו העיקרית של הפרויקט שלנו היא במימוש מערכת הצפנה שיכלול הגנות בחומרה שגורמות לפיזור של האינפורמציה בזמן כנגד התקפות צד אלקטרומגנטיות.

התקפות ערוץ צד אלקטרומגנטיות על מערכת החומרה , מאפשרות לנתח בצורה יעילה מהו המידע הנדגם, ובעזרתו למצוא קשר קורלטיבי בין מפתח ההצפנה הסודי של המערכת לפליטות האלקטרומגנטיות של הכרטיס עליו מושתתת מערכת ההגנה.

להגנה על רכיב נגד התקפות ערוץ צד , ישנן מספר דרכים כגון : מיסוך המידע או תוספת רעשים (הסתרה). אנו נתמקד בהגנה המבוססת בפיזור האינפורמציה בזמן של מערך ההצפנה מסוג AES (עם תוספת הגנה של "מריחת" האינפורמציה בזמן.)

היתרון של הגנה מסוג זה היא שהיא דורשת תוספת שטח קטן יחסית על פני הגנות אחרות.

תחילה, ניישם את צופן AES בשפת תיאור חומרה ע"ג כרטיס פיתוח של חברת Xilinx ונאמת את פעולת המערכת. לאחר מכן , נמדוד וננתח את האותות הנפלטים מתוך אלגוריתם ההצפנה AES לא מוגן לאחר ביצוע הקפות שונות ויזומות (מגוון מודלים של התקפה).

בשלב הבא נמשיך בכתיבת ומימוש אלגוריתם ההגנה המוצע ("מריחת" המידע) , ונבחן שיטות יעילות למימוש במערכת FPGA . נריץ שוב את אותן התקפות אלקטרומגנטיות על מערכת החומרה ,נמדוד וננתח את הנתונים מאותות הפלט בעזרת אלגוריתם עיבוד אות הממומש ב Matlab/Python הן עבור המערכת הלא מוגנת והן עבור המערכת המוגנת לצורך מציאת קורלציה בין הפלט למפתח הסודי.

לבסוף נדגים כיצד המימוש המוגן מסייע בסיכול מטרת ההתקפה, קרי חילוץ המידע הסודי מהמערכת

.Summery :

Our project main goal is implementing a defense system which includes hardware defenses that utilizes temporal spreading of information in order to defend against electromagnetic side attacks. Electromagnetic side attacks on a hardware system can enable us to easily analyze the sampled information. With that information we can find a correlation between the encryption key of the system and the electromagnetic pulses of the card on which our defense system is implemented.

There are several ways in which we can shield a component from side attacks, such as using screened data or adding noise. Our project will focus on implementing an encryption system that includes hardware defenses, an inherit advantage of such a system is the relatively small space added, an AES type of defense system, with our added defense that spreads information temporally.

We will start by implementing the AES (without our temporal defenses) on a ‘Xilinx’ development card using Hardware description language, after which we will measure and analyze the signal produced by the unprotected encryption algorithm, AES, as a result of intentional and varied attacks.

We will proceed by writing and implementing our suggested defense algorithm ("data spreading"), and examining efficient methods in a FPGA system. We will run the same electromagnetic attacks on the hardware system again. Next by analyzing and measuring both collected data sets (of the protected and unprotected systems) we will be able to find a correlation between our encryption key and the output pulses with the use of Matlab/Python.

Finally we will demonstrate how our AES implementation help sabotage the attack, and so prevents extracting the secret info from the system.

מבוא :

מהי קריפטוגרפיה ?

קריפטוגרפיה (תּוֹרַת כְּתִיבַת הַסֵּתֶר בעברית) היא ענף במתמטיקה ומדעי המחשב העוסק במחקר ופיתוח שיטות אבטחת מידע ותקשורת נתונים על רובדיהם השונים, בסביבה פתוחה הנגישה לצד שלישי המכונה " יריב ", או " אויב " פוטנציאלי.
התחום מאגד תחתיו פיתוח ואנליזה של פרוטוקולים המתמודדים בהיבטים שונים של אבטחת מידע בנוכחות צד שלישי. מהם בנוסף לסודיות ,הרשאת גישה ,סיסמה ,הוכחת ידיעה ,פרוטוקול אתגר-מענה ,מנגנוני חתימה דיגיטלית ,חלוקת סוד ,חישוב רב משתתפים בטוח ,אימות זהויות ,מניעת הכחשה ועוד.

תורת ההצפנות (קריפטוגרפיה) כבר קיים שנים רבות עוד מתקופת המצרים בערך מהמאה ה20 לפני הספירה , וכבר אז ניתן היה לראות נסיונות להסתרת מידע באמצעות סוגי צופן שונים. גם בימינו אנו תורת ההצפנות משמשת לאבטחת מידע נגד תוקפים או אויבים פוטנציאלים.
במהלך המאה ה20 ובהמשכה בעקבות התפתחות הטכנולוגיה התפתח גם עולם ההצפנות אשר בתוכו התפתח גם תחום הקריפטואנליזה המתעסק בניתוח וחקר מערכות מידע בכדי לגלות היבטים סודיים של המערכת .

המטרה העיקרית של תחום הקריפטואנליזה היא לגלות מידע רב ככל הניתן על הplaintext (המידע המקורי) בעזרת ניתוח הcipher text (המידע המוצפן).

מהי הצפנה ?

במובן הבסיסי, המונח הצפנה (באנגלית: encryption) מתאר הסתרת משמעותו של מסר קריא באמצעות פונקציה שמקבלת כפרמטר מפתח הצפנה והופכת את המסר לרצף של סימנים בלתי מובן לאיש. מטרת ההצפנה היא להסתיר את המסר , כך שלא יהיה גלוי בפני אף אדם מלבד האדם אליו מיועד המסר .

שחזור הטקסט המוצפן למצבו הקריא באמצעות פונקציה הופכית מתאימה עם מפתח הפענוח, קרוי פענוח (באנגלית : decryption) המונח צופן (Cipher)מתייחס לאלגוריתם הצפנה בדרך כלל במחשב, כאשר קלט האלגוריתם נקרא תַּמְלִיל פָּשׁוּט או טקסט גלוי (באנגלית :Plaintext) ואילו פלט האלגוריתם נקרא תַּמְלִיל מֻצְפָּן או טקסט מוצפן(באנגלית :Ciphertext) .

פעולת אלגוריתם ההצפנה נשלטת על ידי מפתח ההצפנה הסודי הידוע רק לשולח ולמקבל. בשפה העברית משמשת המילה צופן גם כשם עצם לכתב סתר או קוד. שיטות הסתרת מידע שאינן עושות שימוש במפתח הצפנה קרויות סטגנוגרפיה.

בעשורים האחרונים של המאה העשרים ההצפנה עברה מהפך משמעותי מסוג של מיומנות שהיא נחלתם של מעטים, שבעיקרה הייתה אוסף פעולות אד הוק שמסתמכות על התחכום של מפתח הצופן למדע מבוסס היטב שנשען על יסודות תאורטיים מוצקים .

שני אירועים לכאורה לא קשורים שאירעו בשנות השבעים של המאה העשרים בסמיכות רבה, הביאו למפנה העיקרי ותרמו יותר מכל להפיכתה למודרנית כפי שהיא מוכרת כיום; הפצת DES כתקן הצפנה והמצאת RSA .

<u>קריפטוגרפיה מודרנית התפתחה בשני מישורים מקבילים אילו :</u>
<u>הצפנה סימטרית והצפנה א-סימטרית.</u>

<u>הצפנה סימטרית –</u>

הצפנה סימטרית היא הוותיקה ביותר ושורשיה החלו עם ההצפנה הקלאסית לפני מאות שנים. הצפנה סימטרית היא אלגוריתם הצפנה שעושה שימוש באותו מפתח הן להצפנת המידע והן לפענוחו. כלומר בשיטה הסימטרית כדי שניתן יהיה להעביר מידע סודי בין הצדדים המתקשרים, שניהם נדרשים להחזיק באותו מפתח הצפנה .

DES היא דוגמא קלאסית להצפנה סימטרית מודרנית.

הצפנה סימטרית פועלת עם מפתח הצפנה זהה הן להצפנה והן לפענוח ובדרך כלל נדרש להחליפו מעת לעת. החיסרון העיקרי בשיטות הצפנה סימטריות הוא העובדה שיש לשמור את מפתח ההצפנה בסוד ואין לחשוף אותו בשום שלב לאף אחד מלבד המשתתפים הלגיטימיים, עובדה שיוצרת בעיה מהותית הנקראת בעיית הפצת מפתחות .כל זוג מתקשרים אפשרי צריכים לשתף ביניהם מפתח הצפנה סודי נפרד לצורך התקשרות אחת או מספר מוגבל של התקשרויות ולשם כך הם צריכים למצוא דרך להעבירו מאחד לשני בסודיות ולאחסנו במקום מוגן. בתקשורת מודרנית מספר המפתחות האפשריים שיש לנהל מבחינה פרקטית הופך לנטל כבד ולעיתים אינו מעשי. בעיה זו נקראת גם פרדוקס הביצה והתרנגולת כיוון שאם קיימת דרך בטוחה להעביר מפתח הצפנה סודי מלכתחילה הרי שאין צורך בהצפנה כי אפשר להשתמש בה להעברת המסר עצמו, מאידך אם אין דרך כזו כיצד יוכלו להעביר מסרים מוצפנים ביניהם ולפענח אותם אם המתקשרים מעולם לא נפגשו.

הצפנה א-סימטרית -

ההצפנה האסימטרית היא שיטת הצפנה שבה המקבל מכין לעצמו שני מפתחות, אחד הנקרא מפתח פרטי שנשמר בסוד ומשמש לפענוח ואילו השני נקרא מפתח ציבורי המפורסם לכל דורש ומיועד להצפנה בלבד. אף על פי שקיים קשר הדוק בין שני המפתחות למתבונן מהצד אמור להיות קשה מאוד לנחש מהו מפתח אחד בהינתן השני ולהפך. בדרך זו כל אחד יכול להצפין מידע עם המפתח הציבורי של המקבל כי הוא ידוע לכל, אך רק המקבל לבדו מסוגל לפענחו עם מפתח הפענוח המתאים שנשמר בסוד

RSA היא דוגמא להצפנה אסימטרית מודרנית .

המכנה המשותף של כל המערכות האסימטריות בימינו לעומת הצפנה סימטרית, הוא היעילות החישובית.

מרבית האלגוריתמים נדרשים לבצע חישובים אריתמטיים ארוכים מאוד בשל העובדה שהבעיות המתמטיות האמורות ניתנות לפתרון באמצעות אלגוריתמים המסוגלים לתת פתרון בזמן ריצה תת-מעריכי שזה טוב יותר מכוח גס כלומר ניסוי כל האפשרויות או מחצית מהן במקרה הממוצע עד לגילוי מפתח ההצפנה. כדי לפצות על כך יש צורך להגדיל את מפתחות ההצפנה בהתאם כך שסיבוכיות הניסיון לשבור את ההצפנה עם מיטב האלגוריתמים הידועים תהיה מעבר ליכולת המחשוב הנוכחית. מסיבה זו השימוש בהצפנה אסימטרית בדרך כלל מוגבל לכמות מועטה של מידע והוא משולב במערכת היברדית עם הצפנה סימטרית לניצול מיטבי של היתרונות שבשתי השיטות. בדרך כלל מפתח ציבורי משמש להעברת מפתח הצפנה סודי, כאשר את ההצפנה בפועל מעדיפים לבצע עם אלגוריתם סימטרי מהיר כמו AES.

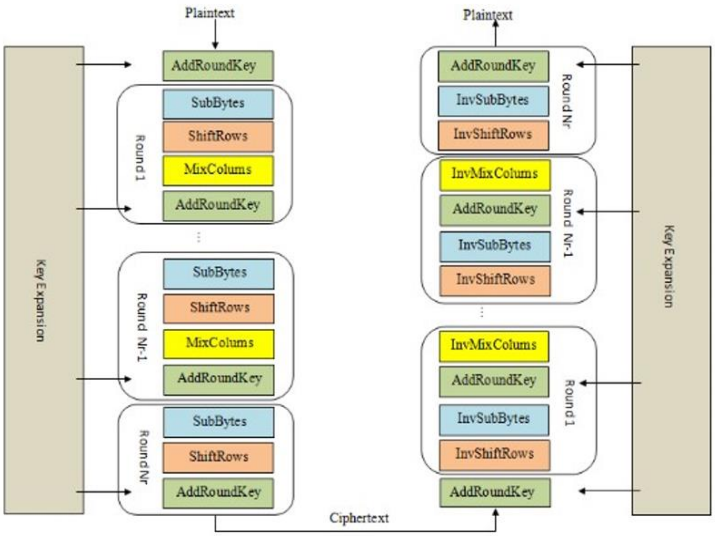
בעקבות התקדמות הטכנולוגיה והופעתם של התקפות קרטוגרפיות משופרות היה צורך למצוא אלגוריתם מתקדם יותר שיאבטח את המידע, לכן במשך כמה שנים נפתחה תחרות פתוחה ע"י המכון הלאומי לתקנים וטכנולוגיה (NIST) למציאת אלגוריתם שיענה לדרישות, האלגוריתמים השונים נבדקו היטב ע"י מומחים רבים. האלגוריתם שנבחר להחליף את DES נקרא צופן ריינדל (Rijndael) לאחר שאומץ ע"י ה-NIST הוחלף שמו לתקן הצפנה מתקדם AES.

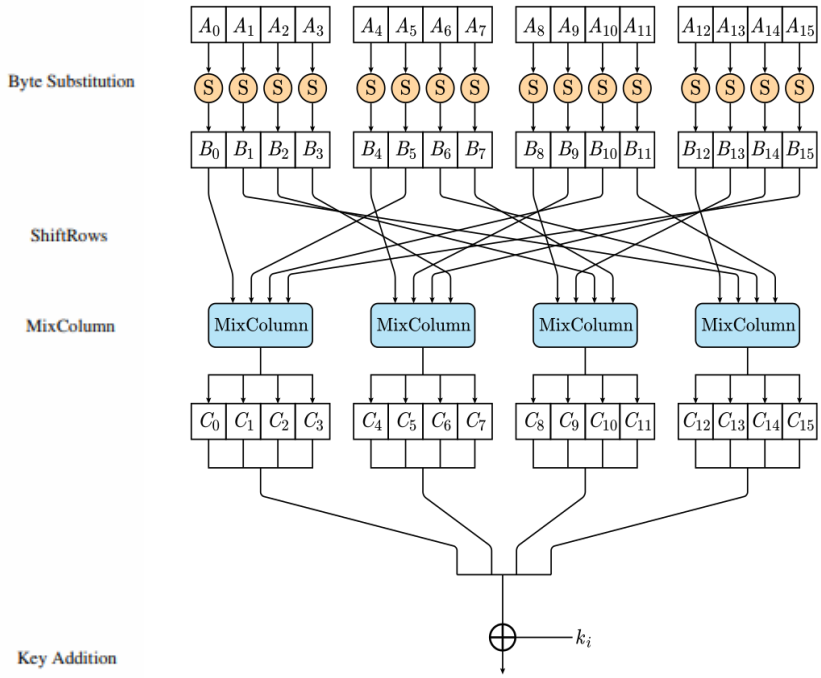
AES הוא צופן בלוקים סימטרי הנמצא בשימוש מעשי כיום בעולם כולו.

האלגוריתם ידוע כאלגוריתם בטוח, כלומר נכון לימינו לא נמצאה התקפה באמצעות תוכנה המסכנת את השימוש בצופן באופן מעשי.

- AES

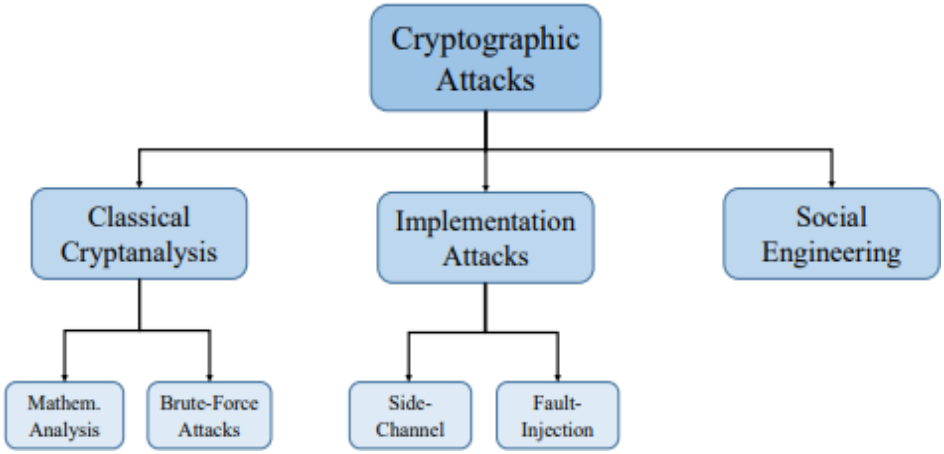
תקן הצפנה מתקדם (באנגלית: Advanced Encryption Standard) או בקיצור AES הוא צופן בלוקים סימטרי שאומץ על ידי המכון הלאומי לתקנים וטכנולוגיה (NIST) של ארצות הברית כתקן הצפנה רשמי שהתקבל בעולם כולו, להצפנת נתונים מאסיבית. AES או בשמו המקורי ריינדל (Rijndael)פותח על ידי הקריפטוגרפים הבלגיים יוהאן דאמן ווינסנט ריימן והוצע במהלך פרויקט בחירת התקן שאורגן על ידי NIST בשנת 2000. לאחר שזכה בתחרות אומץ על ידי ממשלת ארצות הברית באופן רשמי להצפנת נתונים מסווגים עבור הממשל והחליף בכך את קודמו DESשיצא לאור ב-1977. אלגוריתם AES נמצא בשימוש מעשי נרחב בכל העולם הן בתוכנה והן בחומרה וידוע כאלגוריתם בטוח.





איור 1 - ארכיטקטורת הAES

סיווג ההתקפות קריפטוגרפיות השונות



איור 2 - סוגי התקפות קריפטוגרפיות

אלגוריתם הAES נמצא חסין נגד התקפות מסוג Brute Force ואלגוריתמים מתמטיים שונים , אשר למעשה מבצע ניסיון חילוץ המפתח הסודי באמצעות מעבר על כל הצירופים השונים והתקפות מתמטיות ע"י יישום אלגוריתמים מתמטיים שונים .

שיטה נוספת לתקיפת מערכות מוצפנות היא שיטת Implementation Attack שבתוכו נמצאות תת נושא התקפות ערוץ צד המבוססות על איסוף מידע על החומרה שעליה יושב המידע המוצפן באמצעות חקירת החולשות החומרתיות והתאורטיות באלגוריתם ההצפנה של המערכת .
שיטת התקפה זו מחולקת להתקפות אקטיביות ולהתקפות פסיביות .

Fault-injection (נקרא גם התקפה אקטיבית) היא התקפה אקטיבית אשר תחת Implementation Attack המבצעת תקיפה פיזית של המערכת בכדי לגלות את המפתח הסודי תוך שיבוש תהליך עבודת המערכת כמו :
שינוי תדר שעון המערכת, התקפה בעזרת קרינה אלקטרו-מגנטית והזרקת קפיצות מתח ושגיאות למערכת שלאחר ניתוח מוצא המערכת ניתן לגלות את המפתח הסודי.
החיסרון של שיטה זו הוא שהיא לא מתבצעת באופן חשאי , אלה מותירה עקבות מאחורי התוקף.

בפרויקט שלנו , אנו נחפש , נחקור וניצור מערכות הגנה מפני התקפות שונות כמו:

Implementation Attack - DPA / CPA הכוללות התקפות ערוץ צד (באנגלית Side channel attack). התקפת ערוץ צד היא התקפת חומרה פאסיבית, המטרה היא לנצל חולשות במערכת שמדליפה מידע שלא ביודעין העוזר בחילוץ המפתח הסודי.

בפרויקט שלנו אנו נתמקד בהגנות כנגד התקפות CPA ו – DPA .

התקפות CPA ו- DPA

ניתוח הספק דיפרצניאלי : DPA- Differential Power Analysis

DPA היא אחת משיטות ההתקפה החזקות ביותר שקיימות. אפילו שקריפטוגרפים ידועים בכך שאלגוריתמים קריפטוגרפיים יכולים לדלוף מידע על המפתח ע"י קרינה אלקטרומגנטים כלשהי, אבל לא הרבה היה ידוע בספרות המקצועית על סוג ההדלפה שהתרחשה ועל איך התוקף יכול לנצל אותה. ב1998 פול קוצ'ר, ג'ושעה וג'ף בנג'ימין הציעו את הDPA ומאז מאות עבודות מחקר נכתבו ושיכללו את שיטת הDPA הבסיסית. פיתחו תאוריית בסיס טובה ואולי יותר חשוב מהכול פיתחו אמצעי נגד ליישומים שעובדים בניגוד לDPA. אלגוריתמים סמיטרים ואיסמטרים יכולים להיות מותקפים ע"י DPA. אבל הרבה יותר קל להגן על אלגוריתמים איסמטרים כנגד DPA באמצעות אמצעים מתמטיים כגון מסכות ואקראיות. מנגד יישום אמצעים נגד למעפנחי בלוקים שיכולים לעמוד כנגד התקפות DPA הם די קשות.

התקפות DPA מבוססות לרוב על צפייה בצריכת האנרגיה של יישום קריפטוגרפיה או קרינה אלקטרומגנטית, איפה שאנחנו משתמשים בAES כדוגמא. הצורה הראשונה היא מה שנקראת DPA, השנייה מכונה DEMA או ניתוח אלקטרומגנטית דיפרצניאל. עבור DEMA אנטנה קטנה ניצבת ליד המכשיר בדרך כלל במרחק כמה מילמטרים של השבבים שמפעילים את הAES. מכאן אפשר לראות את הDMEA כאמצעי אלחוטי למדיית צריכת האנרגיה. למרות שאפשר לחשוב ששני השיטות שונות. הם מספקות לתוקף מידע דומה.ההיתרון של ניתוח אנרגיה שלעיתים יותר קל לבצע את המדידה עצמה. מצד שני אות האנגריה מכיל לא רק את המידע מהאלגוריתם הקריטוגרפי אבל גם כל "רעש אחר מהמכשיר". בנוסף מערכת הכוח הפנימית עובדת כפילטר חלש המחסר מידע מועיל אודות מפתח הקריפטו. באופן מעשי ניתן להשתמש לרוב בשני השיטות.

קורלציית ניתוח הספק : CPA - Correlation Power Analysis

אפשר לראות את CPA כהכללה של DPA. זה משתמש בגישה יותר אנליטית שמנצלת את המידע שדולף בתוך המפתח. מה שהוצע ב2004 ע"י אריק ברייר, כריסטפר קלייבר ופרנסיס אוליבר. נתחיל עם תיאור ברמה גבוהה של CPA, שמציג את אותם נוסחאות מתמטיות באופן עוקב.

בDPA קלאסי כלומר DPA הבדלי הדברים מחלק קודם המתקיף חזה ביט אחד מערך תלוי מפתח. בהערות שלנו הערך נקרא b0. כדי לאשר את תחזיותיו הוא מסתכל על השינויים בצריכת האנרגיה שנגרם על ידי הביט. גישה זו הנעשית ע"י הDPA אי לא האופטמילית הביטים האחרים של הb0 גם מכילים מידע אודות המפתח איך לא נעשה בהם שימוש.
הרעיון מאחורי CPA הוא שהערך של בייט שלם תואם לצריכת האנרגיה. באופן יותר מדוייק המתקיף מעלה תחזיות בנוגה לצריך האנרגיה שמבוססות על היפותזות שהוא מחשב, ובודק עד כמה הם תואמות עם צריכת האנרגיה שמתרחשת למעשה.

מטרת הפרויקט :

המטרה העיקרית של הפרויקט היא לממש מערכת הצפנה מסוג AES על מערכת חומרה, תוך מימוש הגנות המאפשרות מריחה של עכבת המידע, לבצע מהלך התקפה על מערך החומרה לחילוץ המפתח הסודי ובחינה של כמות המידע הזולג לאחר המימוש והשוואה למערכת לא מוגנת.

מדדי ביניים ויעדי הפרויקט

1. מימוש של מערכת הצפנה על רכיב FPGA או Arduino – לימוד רכיב FPGA ומימוש מערכת הצפנה סימטרית בסיסית מסוג AES בדיקה ואימות של המימוש שבוצע. תכנון הגנות בעת המימוש שמונעות אפשרות לביצוע התקפות ערוץ צד.
2. מדידות של עכבות הפליטה האלקטרומגנטית תוך כדי הרצה של מילות הצפנה על רכיב הFPGA ושמירה של האותות הנמדדים באמצעות החיישן בזיכרון האוסילוסקופ.
3. אנליזות של האותות הנמדדים – כתיבה של קוד Python/MATLAB המסוגל לטעון את האותות הנמדדים ולבצע עיבוד לאותות שנמדדו. חישוב הקורלציה בין האותות הנמדדים ומציאת המפתח בעל הקורלציה הגבוהה ביותר.
4. מימוש אלגוריתם פיזור האינפורמציה בזמן ומדידה חוזרת של האותות ובדיקה של יכולות התקיפה לאחר המימוש.

על מנת לממש מערכות מסוג זה , נצטרך לעבור דרך צעדי המחקר :

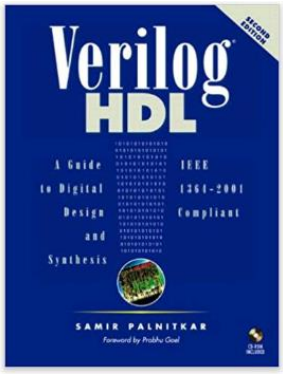
- ללמוד ולהכיר איך התקפות מסוג DPA וCPA עובדות.
- יצירת מערך בדיקה למדידת האות הנפלט , מדידת המידע שזלג למתקפות אלה,
- שמירת האותות עבור מילות הצפנה שונות וניתוחן .
- אנליזות של האותות הנמדדים.
- לימוד וחקר מערכות הגנה רכיבי FPGA ומערכות הגנה כגון AES .
- ללמוד השמת "מריחת" האינפורמציה באמצעות מערכות הגנה חומרתיות/תוכניתיות .
- חקירה מתודולוגיה ובניית הגנה משלנו.
- מימוש הגנה (כגון: אלגוריתם הצפנה)על רכיב הFPGA ואימותו במנגנון .
- לנתח כמות המידע שזלג לאחר ממוש הגנות ולהשוואות הם המדידות הראשונות ולהסיק מסכנות מכך.

לאור העובדה שבחרנו בהתמחויות זהות , ייתכן שחלוקה זו תשתנה במהלך העבודה .

מדדים להצלחת הפרויקט

- יישום בפועל של אלגוריתם ההגנה מפני חילוץ המפתח הסודי באמצעות התקפת ערוץ צד ואימות המנגנון.
- הארכת זמן ביצוע ההתקפה של התוקף ככל הניתן .
- ניתוח ההתקפה היזומה שלנו על אלגוריתם ההגנה

סקירה ספרותית:



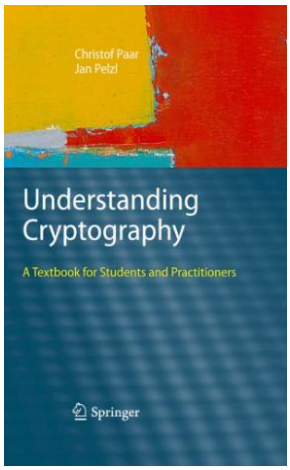
- ספר העוסק בלימוד שפת Verilog.

הפרק הראשון של הספר יקנה לנו כלים בסיסיים שבעזרתם נוכל להיעזר לשימוש בשפת Verilog .

הפרק השני כולל בתוכו נושאים מתקדמים יותר כגון בניית מערכי בדיקה מתקדמים בסימולציה (test-bench), מידול ברמת מפסקים ולוגיקה לסינתזה.

Samir Palnitkar," A Guide to Digital Design and Synthesis, Second Edition "

,21 April 2003



- הספר והקישור להרצאות עוסק במתן רקע תאורטי בנושא קריפטוגרפיה וקריפטואנליזה.

בספר נוכל למצוא מידע אודות סיווג שיטות קיימות לפיצוח צפנים, הגדרת צפנים סימטריים וכו.

יתרה מזאת הספר מכסה באופן נרחב את הנושא של מעבר בין צפנים פשוטים יותר לצופן שעליו מבוסס הפרויקט שלנו AES, הספר כולל בתוכו תיאור וחקירת מבנה האלגוריתם. בנוסף בספר קיימים תרגילים שונים העוסקים בהבנת האלגוריתם.

Prof. Dr.-Ing. Christof Paar and Dr.-Ing. Jan Pelzl, " Understanding Cryptography", 2010

- קישור להרצאות של Christof Paar :

https://www.youtube.com/channel/UC1usFRN4LCMcfIV7UjHNuQg/videos

- קישור לספר נוסף על התקפות צד וסיווגיהן :
- https://www.emsec.ruhr-uni-bochum.de/media/attachments/files/2015/09/IKV-1_2015-04-28.pdf
- אלגוריתם AES בפירוט

https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.651

Implementation of Cryptographic Schemes 1 Prof. Christof Paar Chair for Embedded Security Ruhr University Bochum

כמובן , שבמהלך הפרויקט נחקור ונקרא מאמרים נוספים , בנושאים המוזכרים לעיל ובנושאים נוספים שנקלע אליהם במהלך הלמידה .

סקר שוק:

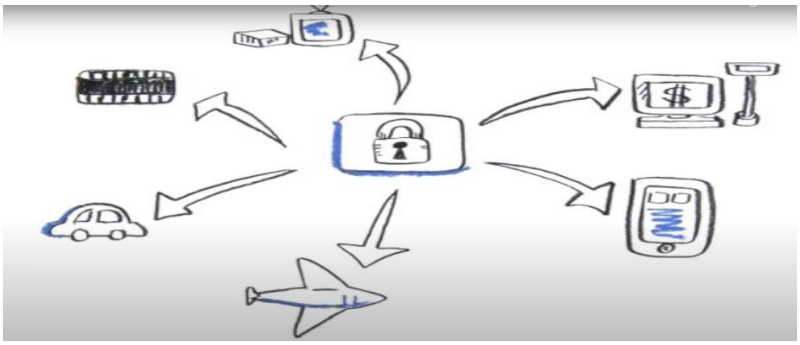
Rambus
Data · Faster · Safer

היא חברה מציעה לצבאות, ממשלות וכל שאר לקוחותיה פתרון להתקפות ערוץ צד. בעזרת Rambus DPA(ניתוח כוח דיפרנציאלי) מגן ליבות חומרה קריפטוגרפיות עמיד נגד התקפות ערוץ צדדיות שונות המנצלות עיצובים קריפטוגרפיים לא מוגנים. . עם ארכיטקטורה בטוחה ונוחה לשימוש , עם טכנולוגיה עצמאית ומערכת הגנה ,בנוסף בעלת מערכת התקפות מובנות מסוג : DPA , DEMA , CPA ו FIA (Fault Injunction Attacks) לצורך בדיקת עמידות נגד התקפות השונות .

התקפות קריפטוגרפיות מסוג DPA התגלו בשנת 1990 , וכבר אז יצאה מערכת הגנה המונעת מהתוקף להשיג את מבוקשו .

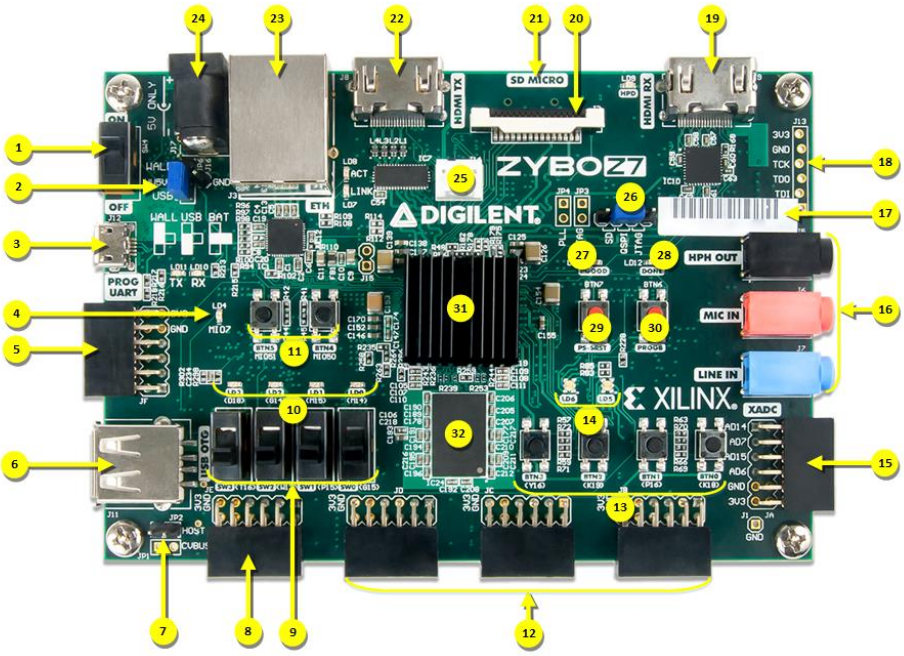
כיום (2020) , קיימות מערכות הגנה רבות המספקות הגנה יציבה ובטוחה ככל הניתן המיוצאות לשוק ע"י חברות שונות וביניהן , חברת Rambus אשר מוערכת כאחת מהמובילות בתחום ושומרת על מידע , הכולל תשלומים , ומידע טלפוני נוסף , של כבערך - $8\ billion$ מכשירים שונים ברחבי העולם .

מערכת ההגנה של *Rambus* מספקת יישום נוח הנגיש למגוון רחב של מכשירים מסוגי שונים.



כמובן , שכיום קיימות שיטות מגוונות לשמירת המידע לרכיבי FPGA שונים .
בפרויקט שלנו נחקור את שיטות ההגנה השונות בכדי ליצור שיטת הגנה.

תכן ראשוני :



איור 3 - רכיב FPGA Xilinx Digilent Zybo Zynq-7000 ARM

FPGA או Field-Programmable Gate Array הוא סוג של מעגל משולב ,אשר ניתן להגדיר ולשנות את תפקודו לאחר הייצור ,בתהליך תכנות הדומה להתקנת תוכנה .הגדרת תפקוד הרכיב מתבצעת על ידי הזנה של תרשים לוגי או על ידי שפות לתיאור חומרה של מעגלים כמו VHDL או VERILOG . את הפרויקט נבצע ע"ג הכרטיס Xilinx Digilent Zybo Zynq-7000 ARM

ה Digilent Zybo Z7-הוא התוספת האחרונה לקו Zybo הנפוץ של פלטפורמת מערכת על שבב Zybo Z7 . ה Zybo Z7-מקיף את ה Zynq®-עם מערך עשיר של אלמנטים היקפיים של מולטימדיה וקישוריות, כדי ליצור מחשב לוח-יחיד נפלא, עוד לפני ששוקלים את הגמישות והעוצמה המוספות על ידי ה .FPGA-מתכנן חומרה ומפתח תוכנה יכולים לשלב בצורה חלקה פונקציונליות FPGA כמו תכנון שערים לוגים רכיבי חומרה ושימוש בPLL וכולי .CPU-מערך מאפייני יכולת-וידאו של ה Zybo Z7- כולל מחבר Pcam תואם ,MIPI CSI-2, כניסת HDMI יציאת HDMI ורוחב פס DDR3L גבוה. צרוף חומרה נוספת נעשה קל על ידי מחברי ה -Pmod של ה Zybo Z7- ,שמאפשרים גישה לקטלוג של Digilent של יותר מ-70 לוחות היקפיים Pmod, כולל בקרי מנוע, חיישנים, צגים ועוד. כמובן שבפרויקט שלנו אנו נשתמש רק בחלק מהרכיבים למימוש המערכת על גבי הכרטיס .

בשלב הראשוני ניישם את צופן AES בשפת תיאור חומרה ע"ג סביבת העבודה Xilinx Vivado® Design Suite ונבין את תוצריה ואופן השימוש בה.

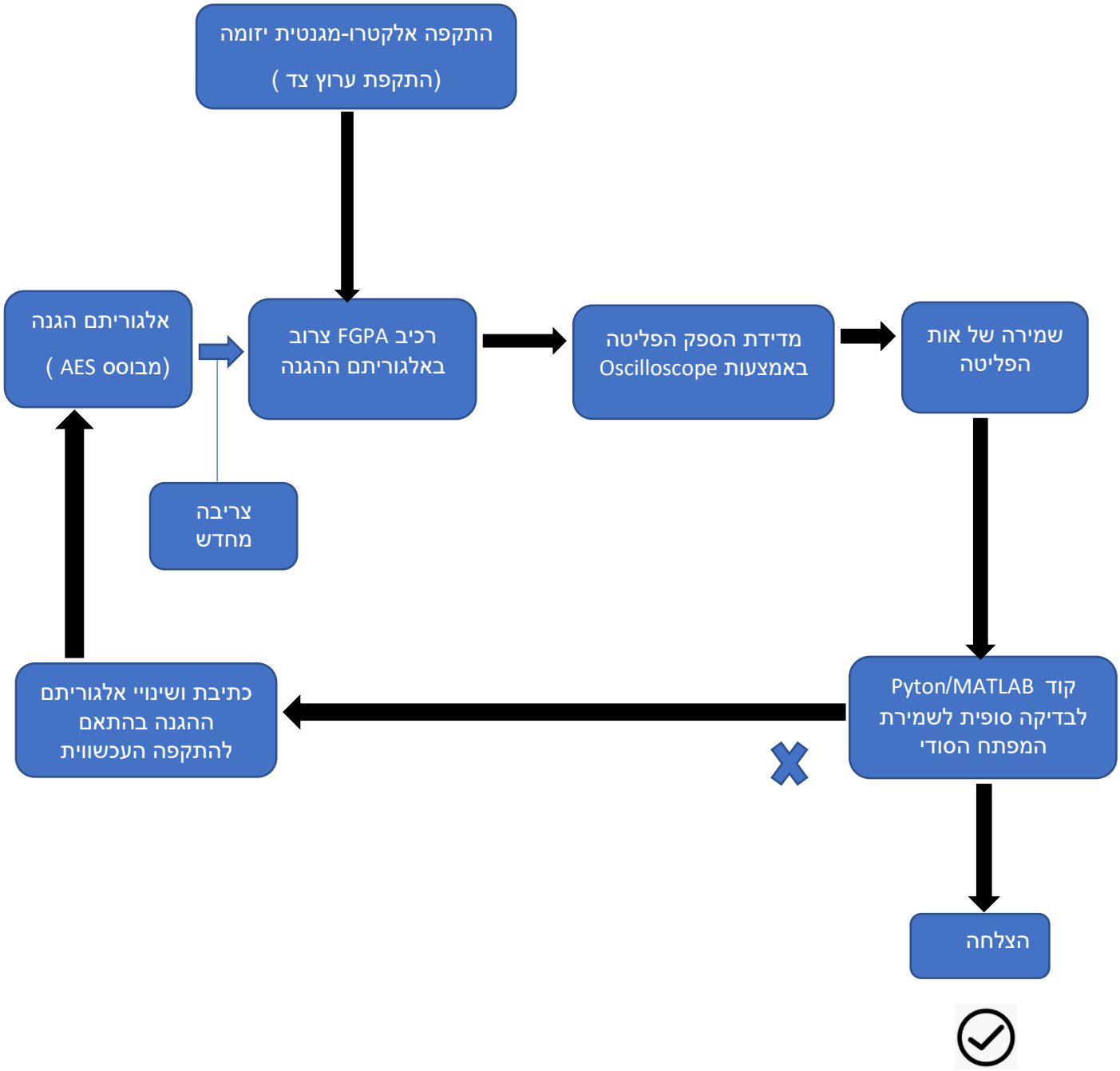
בשלב הבא נבצע בדיקות כדי לוודא שמערכת ההצפנה עובדת בצורה תקינה באמצעות כתיבת test bench מתאים , ולאחר אימות תקינות מערכת ההגנה נוכל לבצע סינתזה וצריבה על לוח ה FPGA.

לאחר מכן , בעזרת עיבוד אותות והשימוש Matlab/Python לצורך מציאת קורלציה בין הפלט למפתח הסודי , נמדוד וננתח את האותות הנפלטים מתוך אלגוריתם ההצפנה AES הטהור לאחר ביצוע מס של התקפות שונות ויזומות .

לאחר ניתוח המידע נמשיך בכתיבת ומימוש אלגוריתם ההגנה שלנו ("מריחת" המידע) , נחזור ונבצע בדיקות כדי לוודא את תקינות מערכת ההצפנה באמצעות כתיבת test bench מתאים , ולאחר אימות התקינות נוכל לבצע סינתזה וצריבה על לוח ה FPGA. נריץ שוב את אותן התקפות יזומות על מערכת החומרה ,נמדוד וננתח את הנתונים מאותות הפלט בעזרת עיבוד אותות והשימוש Matlab/Python לצורך מציאת קורלציה בין הפלט למפתח הסודי בתקווה שלא תיווצר אחת .

נשווה בין ביצועי מערך ההגנה AES הטהור לעומת מערך ההגנה AES עם תוספת מערך ההגנה שלנו .

דיאגרמת בלוקים



דיאגרמת בלוקים – הסבר

לאחר מימוש מערך ההגנה וצריבתו על רכיב FPGA , אנו מבצעים התקפת ערוץ צד יזומה בה אנו מחפשים את המפתח הסודי הנמצא בתוך הרכיב .
לאחר ביצוע ההתקפה , אנו מודדים את ההספק הנפלט מהמערכת שלנו , שומרים את המידע על האותות הנפלטים ומבצעים אנליזה לאותות הנדגמים בעזרת קוד MATLAB או PYTHON בכדי לבדוק כי מערך ההגנה אכן מגן מפני התקפות והמפתח הסודי לא התגלה.
את ההתקפה נבצע גם על המערכת נטולת תוסף ההגנה בכדי לבדוק את שיפור ההגנה.
לאחר מכן , נחפש התאמה בין ה"מפתח הסודי" הנלקח מההתקפה היזומה , אל מול המפתח הסודי עצמו במערכת המוגנת והלא-מוגנת.

במידה ולא קיימת התאמה , הרי שהתוקף קיבל מידע מוטעה ולא הצליח לחלץ את המפתח הסודי אותו חיפש מלכתחילה.

אם קיימת התאמה בין השניים הרי שהתוקף קיבל את מבוקשו וחילץ את המפתח הסודי.
במקרה זה , אנו חוקרים את המקרה ואת האותות הנפלטים , מוסיפים את השינויים לקוד של MATLAB או PYTHON , ומשנים בהתאם את מערך ההגנה ונבצע התקפות חוזרות ונשנות מאותו הסוג , עד שנוודא שהצלחנו לשמור על המפתח.

האתגר ההנדסי בפרויקט :

הוספת הגנות הגורמות למריחת (פיזור) האינפורמציה בזמן , תוך שימוש בטכנולוגיית RandomClock שאותה מצטרך ללמוד, לחקור, לממש ולהתמודד מול אחת הסכנות הגדולות של שימוש במס שעונים העלול לגרום לבעיות טיימינג במהלך יישום הפרויקט .
טכנולוגיה זו תמומש עם אלגוריתם ההצפנה AES כך שלמעשה נקבל את מערכת הRandomClockAES אותה נממש על רכיב הFPGA ונבצע אימות למנגון הקריפטוגרפי .
לבדיקת ההגנות נרכיב מערך בדיקה הכולל : בנייה של הגלאי המשמש למדידת האות הנפלט מהחומרה מדידה באמצעות אוסילוסקופ ושמירה של האותות עבור מילות הצפנה שונות, כתיבת קוד מטלב לניתוח של האותות שנשמרו ואישוש חוסר היכולת לחילוץ המפתח הסודי.

חלופות מערכתיות :

כיום בשוק , קיימים מספר רב של רכיבים בהם נוכל להשתמש . מתוך ההצעות , נבחר את הרכיב הרלוונטי והשימושי ביותר לתקיפות מבין הרכיבים הבאים :

- רכיב Altera FPGA של חברת Intel.
- בקר Arduino
- רכיב FPGA Zynq של חברת Xilinx .

בקר Arduino הינו בקר מוגבל יחסית ביכולותיו ולא היעד העיקרי של התוקף ולכן הוא לא רלוונטי לפרויקט שלנו.

לכן נותרנו עם שני רכיבי FPGA של 2 חברות שונות .
אף על פי שאנו יודעים שכמעט ואין הבדל מבחינת השימוש ביניהם (שניהם יספקו את היעדים כמעט בצורה שווה) נבצע השוואה בין השניים ומבדיקות עולה כי :

קריטריון / חלופה מערכתית	Altera	Xilinx	Arduino
יכולת תקיפה אלקטרומגנטית	10	10	10
בעל יכולת התגוננות מפני תקיפות ערוץ צד	7	7	4
בעל נתח שוק אשר רלוונטי לתקיפה	10	10	7
דורש מחקר נוסף	6	8	5
סיכום	**33**	**35**	**28**

טבלה 1 – השוואת חלופות מערכתיות

אי לכך , הרכיב הנבחר לפרויקט שלנו ובעל הניקוד הגבוה ביותר , הינו רכיב FPGA Zynq של חברת Xilinx .

חלופות טכנולוגיות :

כיום ישנם מספר דרכי התמודדויות לתקפות ערוץ צד.

אנו מציגים מערכת הצפנה מסוג AES על מערכת חומרה , הוספת הגנות בחומרה שגורמות למריחת האינפורמציה בזמן.

יתרונות :

- ההגנה מתבצעת על הרכיב.
- מגנה על מספר רב של התקפות ערוץ צד כגון התקפה אלקטרומגנטית התקפת ניתוחי הספק וכו.

חסרונות:

- רכיבים שונים עובדים שונה (מבחינת: תדרים, זמנים) ולכן עלולים לדרוש כיול לרכיבים מסוגים שונים לצורך התאמה.
- גודל הזיכרון במיקרו-בקר מוגבל (כאשר משתמשים במיקרו בקר) .
- מימוש לא טריוויאלי על הרכיב.

מיסוך מידע

יתרונות:

- הגנה מתבצעת על רכיב.
- מתאימה לטכנולוגיות שונות.

חסרונות:

- מאוד מורכב למימוש.
- גודל הזיכרון במיקרו-בקר(כאשר משתמשים במיקרו בקר) מוגבל ולכן לא ניתן לבצע הצפנה בעלת מספר גדול של סיביות קרי הגבלה בהצפנה.

מחולל רעשים

יתרונות:

- לא מצריך שימוש בזיכרון הרכיבי .
- מתאים בצורה אוניברסלית מספר רב של כרטסים

חסרונות:

- הגנה נמצאת כתוספת חיצונית שתופס מקום על רכיב.
- רעשים העלולים לפגוע במידע שאנו מעברים.
- אחרי מספר רב של התקפות עלול הרעש להפוך ללא רלוונטי.

הצפנה ומימוש על גבי כרטיס FPGA

בצופן ריינדל , רוב הפעולות בצופן הן פעולות אלגבריות על בתים המייצגים איברים של השדה $GF(2^8)$. הסיבות לבחירה זו הן שכל איבר בשדה ניתן לייצוג על ידי בית אחד במחשב ובזכות תכונת האיזומורפיות .למען יעילות נשתמש בייצוג פולינומי רגיל, כאשר $a0$ מייצג את המקדם החופשי או הראשון , ו $a7$ נקרא המקדם המוביל :

$$\boldsymbol{A(x) = a7x^7 + a6x^6 + \cdots + a1x + a0}$$
$$\boldsymbol{a_i \ £ \ GF(2) = \{0, 1\}}$$

נוסחה 1 – ייצוג פולינומי

לבסוף , פעולת הכפל המודולרי היא :

$$\boldsymbol{C(x) \stackrel{\text{def}}{=} A(x) * B(x) mod(p(x))}$$

כאשר : $P(x) = X^8 + X^4 + X^3 + X + 1$

נוסחה 2 - פעולת הכפל המודולרי

בהמשך להסבר על אלגוריתם ה-AES לעיל , נפרט על ההצפנה של האלגוריתם:

תיאור האלגוריתם –
באלגוריתם שלנו אנו ממשים את עקרונו של צופן ריינדל . זהו צופן בלוקים איטרטיבי בעל מפתח ובלוק המשתנים בגודלם הניתנים להגדרה ללא תלות אחד בשני; 128, 192 או 256 סיביות (בתקן ההצפנה המתקדם גודל הבלוק נקבע ל-128).

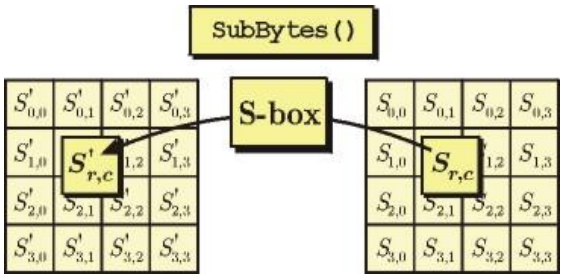
rounds	block size	key size	
10	4	4	AES-128
12	4	6	AES-192
14	4	8	AES-256

טבלה 2 - סבבים וגדלים אפשריים באלגוריתם AES

בכל סבב טרנספורמציות המצב כוללות ארבע פונקציות שונות , למעט בסבב האחרון בו לא מתבצעת פעולת הMixColumn :

1. ByteSub
2. ShiftRow
3. MixColumn
4. AddRoundKey

הטרנספורמציה ByteSub :



איור 4 - טרנספורמציית הByteSub

הטרנספורמציה ByteSub היא פונקציית החלפת בתים אי-ליניארית הפועלת באופן סדרתי על כל בתי המצב ללא תלות אחד בשני על פי טבלת החלפה קבועה והפיכה. בניגוד ל DES- ערכי תיבות (S-box) של AES אינם ערכים אקראיים כלשהם שעונים על המאפיינים הדרושים, אלא הם בעלי מבנה אלגברי מובהק. למעשה אפשר לחשב אותם על ידי שילוב של שתי טרנספורמציות. הראשונה היא חישוב הופכי כפלי של הקלט בשדה $GF(2^8)$ מודולו $P(x)$ כאשר אפס ממופה לעצמו כך שאם הקלט לפונקציה הראשונה של S ה הוא : $A_i = (a7\ ,\ \dots\ ,a0)$, הפלט יהיה :

$$B'_i(x) = A_i^{-1} mod\big(P(x)\big)$$

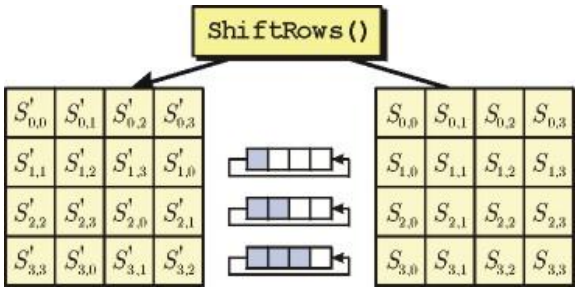
נוסחה 3 – חישוב הופכי כפלי של הקלט בשדה $GF(2^8)$ מודולו $P(x)$.

הפעולה השנייה היא טרנספורמציה אפינית של הפלט מעל השדה F_{2^8} שתפקידה "לטשטש" את המבנה האלגברי הייחודי שלו. הטרנספורמציה האפינית היא הכפלה במטריצה בינארית הפיכה קבועה מסדר 8*8 סיביות וחיבור עם וקטור קבוע באורך 8 סיביות.

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

טבלה 3 – טבלת החלפת הערכים (Sbox)

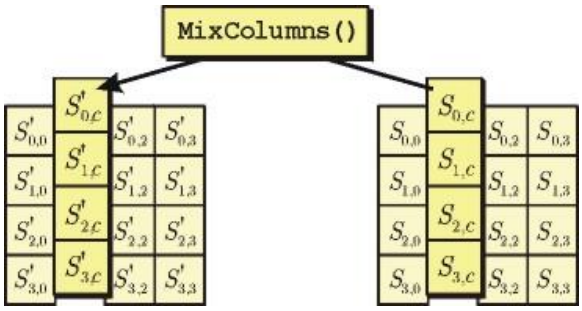
הטרנספורמציה ShiftRow :



איור 5 - תהליך ההזזה המחזורית

הזזת שורות המצב בהזזה מעגלית לפי ערכים שונים. השורה הראשונה נותרת במקומה והשורות הבאות מוזזות לפי היסטים קבועים $C1 , C2, C3$. בהתאמה. ההיסטים תלויים בגודל הבלוק N_b אשר מייצג פונקציה של גודל הבלוק בסיביות חלקי 32 .
הפעולה ההפוכה של טרנספורמציה זו מושגת על ידי הזזה לפי $N_b - C1 , N_b - C2 , N_b -$ $C3$ בהתאמה , כך שהבית במקום j ובשורה i מוסט כעת למיקום :
$(j + N_b - C_i) mod N_b$

הטרנספורמציה MixColumn :



איור 6 - תהליך ערבוב העמודות

עמודות המצב מיוצגות כפולינומים מעל השדה $GF(2^8)$ ומוכפלים mod($X^4 + 1$)
בפולינום קבוע : $C(x) =' 03' * X^3 +'01' * X^2 +'01' * x +'02'$

נוסחה 4 – פולינום המקדמים הקבועים(פולינום עזר)

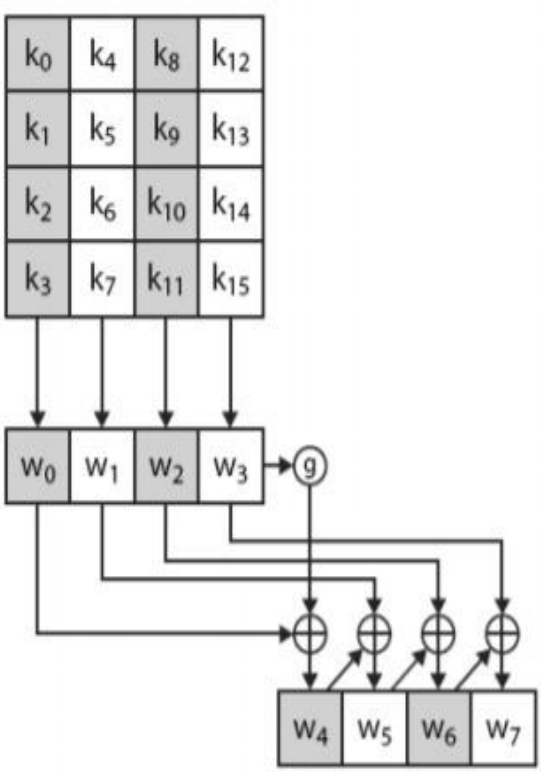
אשר זר לפולינום $X^4 + 1$ ולכן יש לו הופכי .ניתן לתאר פעולה זו בעזרת כפל מטריציוני בצורה הבאה בעזרת שימוש בפונקצית $Gmul$:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

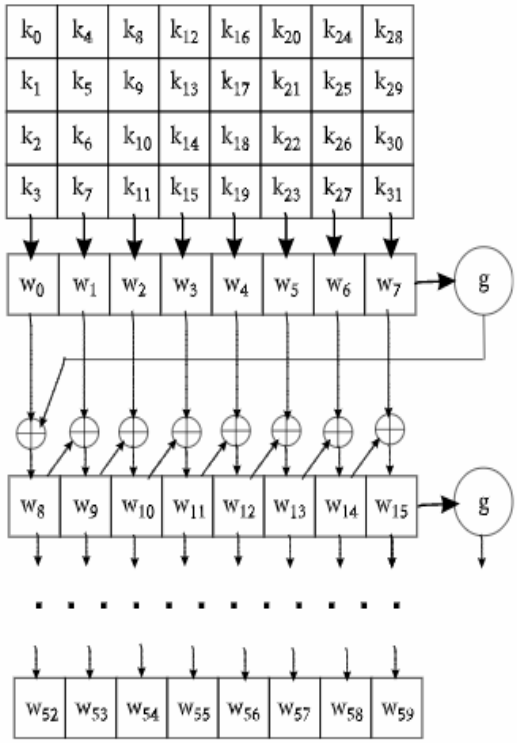
נוסחה 5 – חישוב עזר לערבוב העמודות

ולפענוח מבצעים פעולה דומה. כל עמודה מוכפלת בפולינום $d(x) =' 0B' * X^3 +' 0D' * X^2 +' 09' * X +' 0E'$, שזהו בדיוק האיבר ההופכי של $C(x)$.

<u>תהליך הרחבת המפתח :</u>



AES 128 bit



AES 256 bit

<u>איור 7 - תהליך יצירת המפתח המורחב</u>

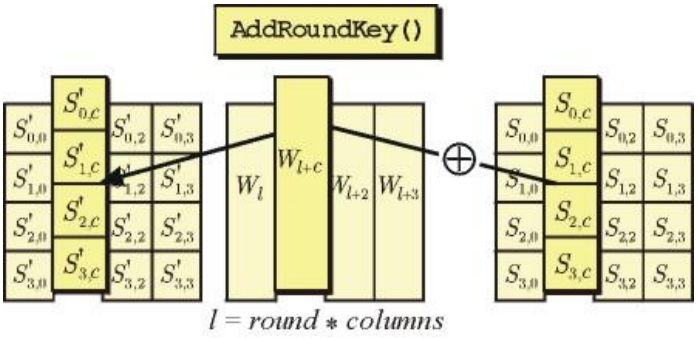
בכדי להצפין את המידע , אנו נצטרך להשתמש במפתח סודי , אשר לכל סבב קיים מפתח משלו , כלומר ניקח את המפתח הקבוע ונבצע עליו את ההרחבה . מפתחות הסבבים הם פונקציה של מפתח הצופן המסופק על ידי המשתמש. סך כל סיביות המפתח המורחב שווה לגודל הבלוק הנבחר כפול מספר הסבבים ועוד אחד. למשל עבור בלוק של 128 סיביות יהיו 1408 סיביות מפתח מורחב, אותם מחלקים
ל-N_b מילים לפי הסדר. למעשה מתקבל מערך חד ממדי של 44 מילים בגודל 4 בתים המסומן $W(N_b * (N_r + 1)$ כאשר N_r מציין את מספר הסבבים הדרושים ותלוי בערכים N_b, N_k . פעולה זו מתבצעת כאשר בכל סבב משתמשים בN_b המילים הבאות . תחילה מחלקים את מפתח הצופן למילים בגודל 32 סיביות אותם מציבים ב N_k- כניסות הראשונות במערך. יתר הכניסות מחושבות באופן רקורסיבי על בסיס ערכים קודמים. כל מילה $W[i]$ היא חישוב XOR עם מילה $W[i - N_k]$ ועבור מילים שמיקומם הוא כפולה של N_k מופעלת טרנספורמציה נוספת שכוללת הזזה מעגלית של בתי המילה בנוסף להחלפה בתיבות ההחלפה וחיבור XOR עם הקבועים $Rcon$.

במקרה זה הפונקציה $SubByte(W)$ מחזירה ארבעה בתים שהם תוצאה של הפעלת תיבות ההחלפה של ריינדל על בתי הקלט. הפונקציה $RotByte(w)$ מחזירה ארבעה בתים שהם תמורה מחזורית של עצמם. דהיינו שינוי סדר הבתים, אם בתי הקלט הם a, b, c, d התוצאה תהיה b, c, d, a. הקבועים $Rcon$ ה*משמשים להרחבת המפתח הם סדרה של עשרה ערכים בגודל 32 סיביות מהצורה

$Rcon_i = (RC_i, '00', '00', '00')$. ערכי RC_i מייצגים את חזקות הפולינום x^{i-1} אזי מתחילים בערך $RC_1 = 1$ ואז מחשבים את שאר הקבועים בהתאם . עבור מפתח של 128 סיביות הקבועים הם : $\{01, 02, 04, 08, 10, 20, 40, 80, 1B, 36\}$

במפתח 256 סיביות משתמשים רק בשבעת הערכים הראשונים.

<u>פונקציית הוספת מפתח :</u>



<u>איור 8 - הוספת המפתח ע"י פעולת XOR</u>

מעדכנים את ערכי כל בתי המצב באמצעות חישוב XOR עם מפתח הסבב (round key) . את מפתח הסבב מפיקים ממפתח הצופן באמצעות התהליך המתואר להלן וגודלו נקבע לפי גודל הבלוק N_b. לדוגמה המערך הדו ממדי של המצב הוא $S_0, c, S_1, c, S_2, c, S_3, c$ כאשר מספר העמודה משתנה $N_b = (0, \dots, N_b)$ ומספר הסבב $N_r = (0, \dots, N_r)$ מחשבים את

$$[S_0, c, S_1, c, S_2, c, S_3, c]\ Xor\ [W_{rN_b+c}]$$

<u>נוסחה 6 – חישוב Round Key לdata לאחר הפעולות בהתאם לRound .</u>

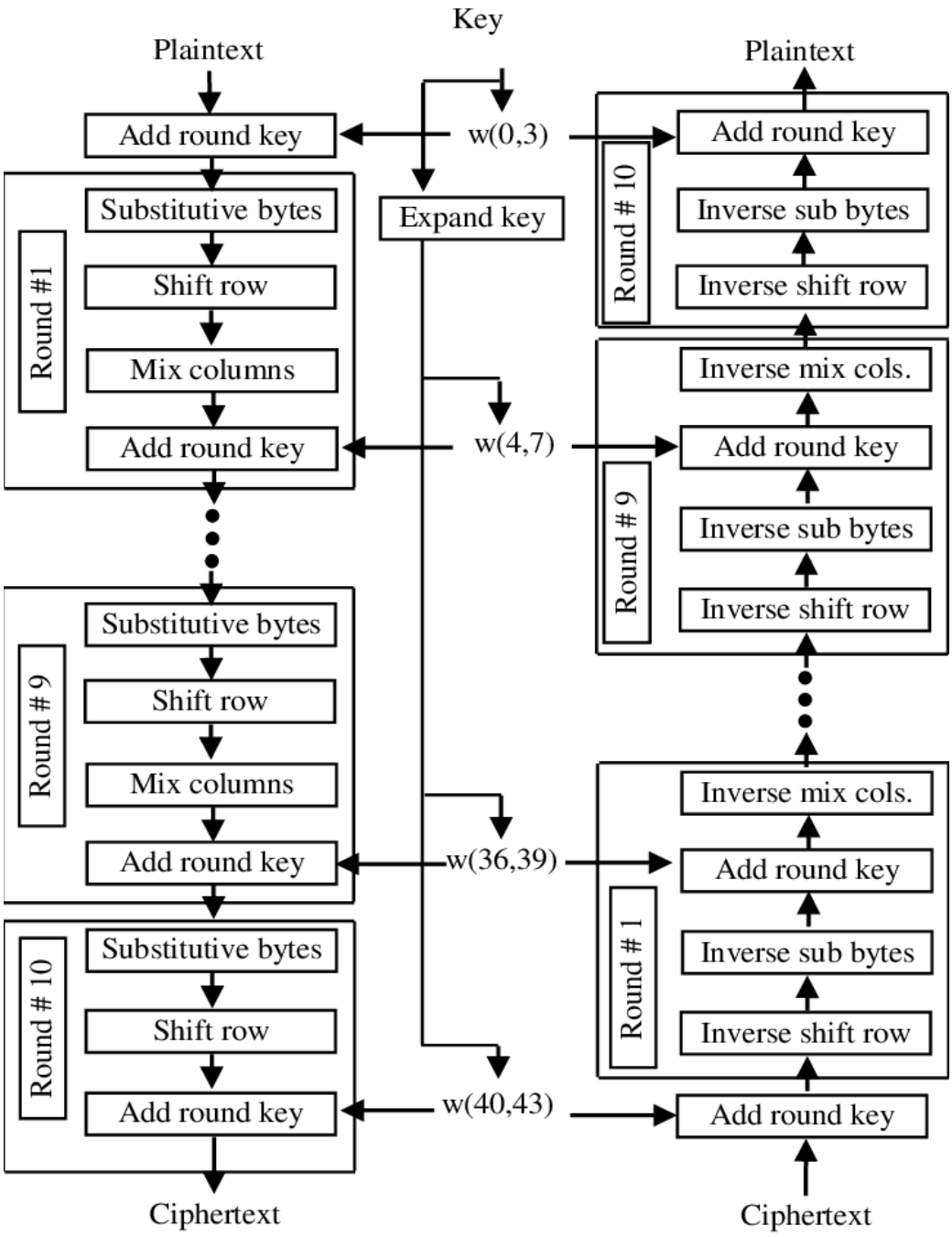
הפעולה ההפוכה בתהליך הפענוח זהה כיוון שהפעולה XOR הופכית של עצמה.

דוגמא לסבב הצפנה :

לוקחים את מטריצת הסבב הראשון נבצע פעולת XOR אל מול המפתח ונעביר אותה דרך ה S-box באמצעות הפונקציה Sub Bytes. למטריצה החדשה שקיבלנו נבצע הזזת שורות באמצעות פונקציית Shift Rows, לאחר מכן מבצעים ערבול עמודות באמצעות הכפלה במרחב גלילאו במטריצה קבועה (התהליך מתבצע באמצעות פונקציית Mix Column) .
לבסוף נבצע פעולת XOR בין המטריצה שקיבלנו למטריצת Round Key הנוכחית , בעזרת פונקציית Add Round Key וכך נקבל את המטריצה לתחילת הסבב השני.

התהליך חוזר חלילה עד הסבב האחרון, כאשר המטריצה האחרונה שנקבל הינה הטקסט המוצפן (ciphertext).

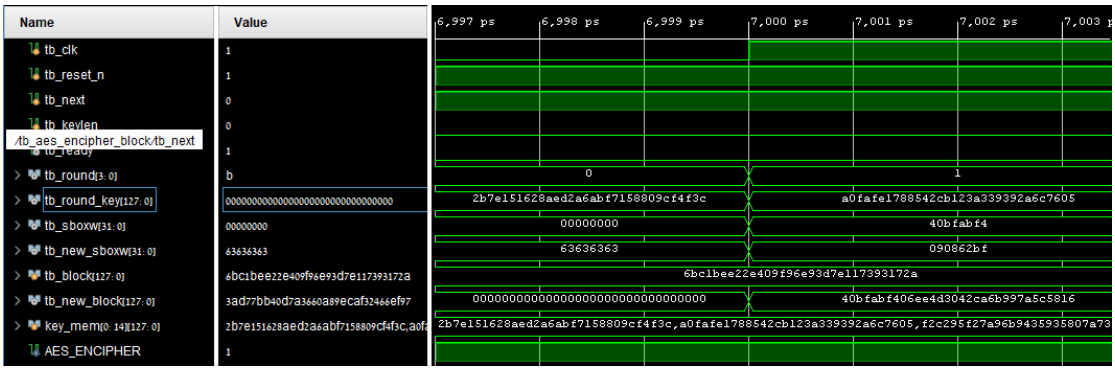
דוגמא ויזואלית לתהליך ההצפנה מופיע בחלק הנספחים (Python outcome)



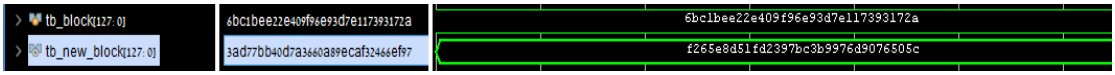
איור 9 - תרשים מלבני המתאר את תהליך ההצפנה של 128b AES (צד שמאל) ואת התהליך הפענוח (צד ימין):

בדיקת פונקציונליות לאלגוריתם הAES הממומש :

הרצנו את הקוד בסביבת ריצה (Testbench) , בכדי להראות את התפוקות והשינויים במהלך הסבבים השונים . נתמקד בשלושה תהליכים חשובים :
הסבב ההתחלתי , סבב 1 (נזכיר סבבים 1-9 מתנהגים באופן שווה) , והסבב הסופי בתהליך ההצפנה .



איור 10 - סימולציית הסבב ההתחלתי - Initial

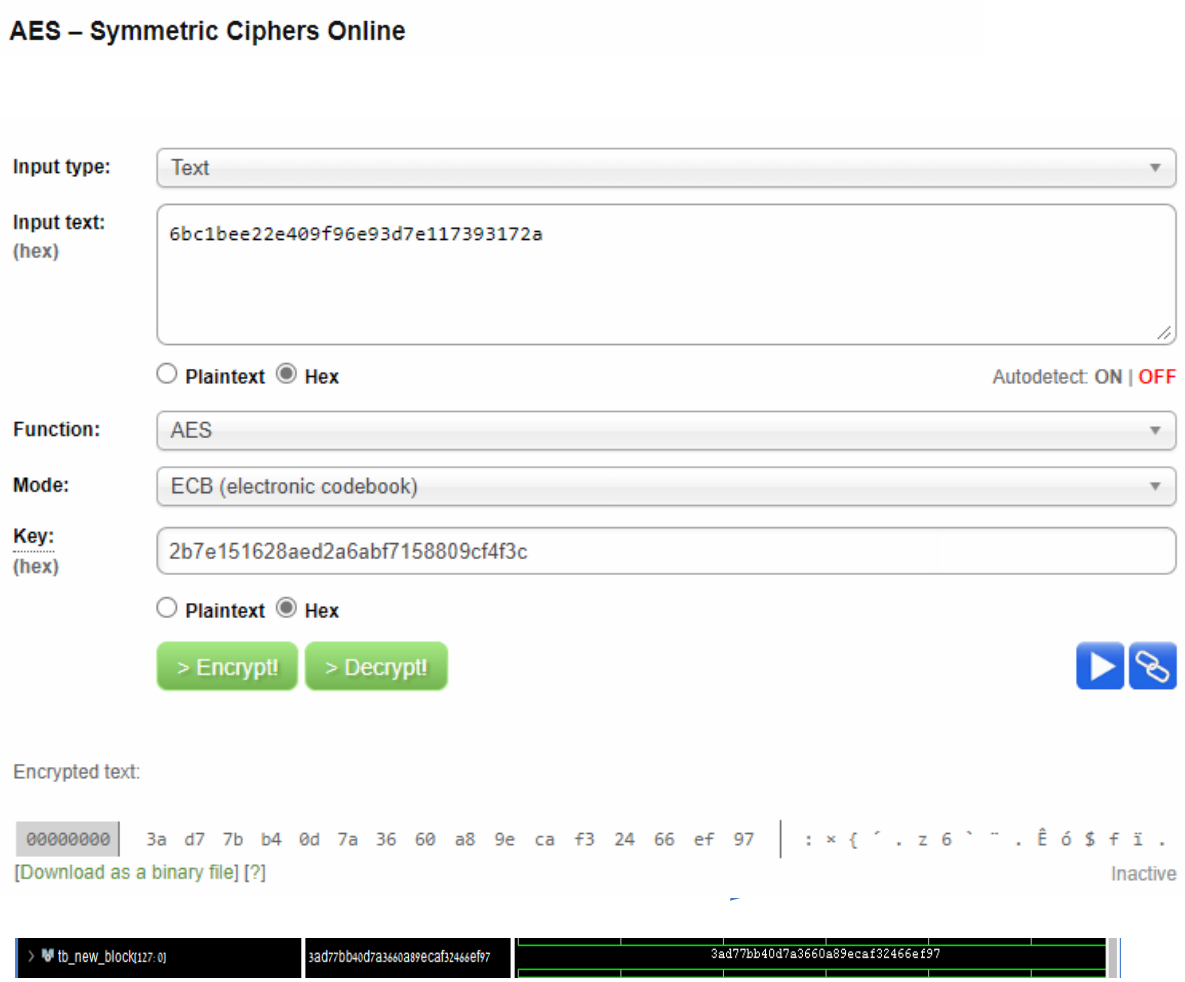


איור 9 תפוקת הסבב ה1 - Round 1 output

איור 12 - הסבב האחרון - Final round

את התוצאות , השוונו עם קוד הכתוב בשפת PYTHON , וכמו שציפינו קיבלנו תוצאות זהות. (ראה את הPLOT של הקוד ואת חישוב התהליך בנספחים) .

בנוסף , הכנסנו את הKEY והDATA אל תוך מחשבון AES וכמצופה קיבלנו תוצאות השוות לתוצאות קודי הVERILOG והPYTHON .



איור 10 - השוואה בין מחשבון הצפנת AES לתוצאת הצפנת קוד Verilog

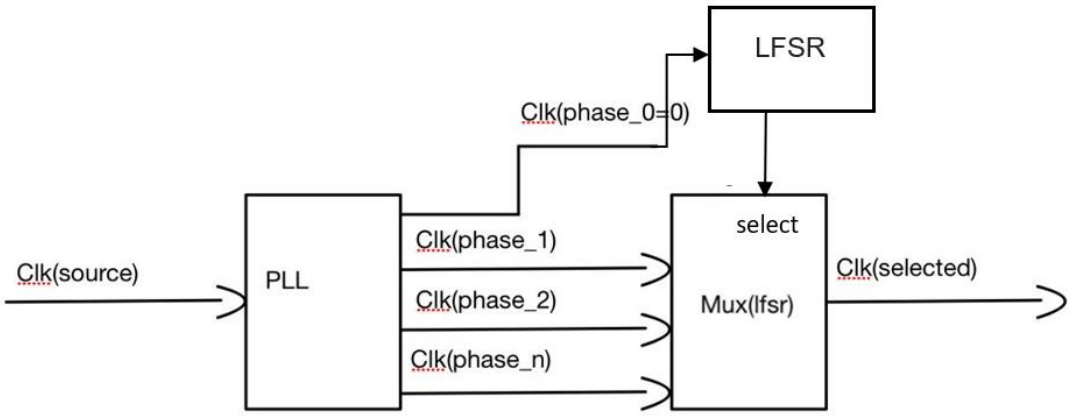
מהתמונות לעיל ומתוצאות המחשבון , נוכל להסיק כי הצפנת הAES אכן מתפקדת היטב . זאת ניתן להבין כאשר הצופן המחושב בתוצאות הסימולציה בתור tb_new_block תואם בדיוק לערך המחושב במחשבון ההצפנה.

מימוש הצפנת AES בעזרת טכנולוגיית Random Clock :

אבטחת יישומי FPGA נפגעת גם בגלל דליפות ערוצים צדדיים, המחייבות שילוב של אמצעי נגד כנגד התקפות ערוצי הצד (SCA) . אמצעי-נגד אלה מפליגים או מדליפים דליפות במטרה להתנגד למדידות דליפות או מסבכים את ההשוואה בין הערכים הנמדדים לעומת הערכים הנמדדים במהלך התקפות בהתבסס על ניתוחים סטטיסטיים.

בכדי להתמודד מול התקפות CPA,DPA ניתן להשתמש בשיטת ה Random Clock (כך למעשה אנו גורמים ל"מריחת זמן" במהלך ההתקפה , כלומר גורם לפיזור של האינפורמציה בזמן)

תכנון ראשוני של Random Clock :



איור 11 – ארכיטקטורה ראשונית של מימוש Random Clock

הסבר כללי על הארכיטקטורה הראשונית של ה-RandomClock :

בשיטה זו , תדר השעון משתנה באופן אקראי לאחר כל מחזור. במילים פשוטות, רוחב הדופק של השעון משתנה באופן פסאודו באקראי באמצעות Mux מסוג LFSR (רגל הSELECT של הMUX)
האחראי להזנת המערכת בשעון אקראי בסוף כל מחזור .

נכניס את תדר השעון המקורי אל תוך רכיב PLL , ובאמצעות שינויי בפאזה נקבל במוצאו אותות שעון שונים אותם נזין אל המערכת . שעון אחד CLK_0 יהיה בעל פאזה 0 בכדי לקבל גם את אות השעון המקורי , זהו יהיה גם השעון השולט על הMUX_{LFSR} .

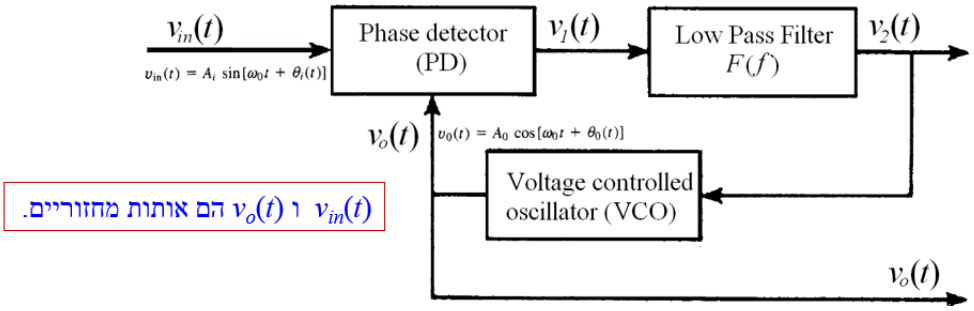
בנוסף , נוציא קו reset אסינכרוני אל תוך רגל הLOCK של רכיב הPLL כך שכאשר אנו נמצאים במצב עבודה תקין , אנו בטוחים שהתבצעה הנעילה . לאחר צריבה התברר שעדיף להכניס reset עצמאי אל הAES , במקום שרגל ה LOCK תשמש כreset ל AES . בצורה זו אנו שולטים על איפוס מערכת הAES.

PLL(Phase Locked Loop):

רקע עיוני –

חוג נעול מופע (PLL - Phase Locked Loop) הוא מערכת בקרה להפקת אות התלוי באופן כלשהו במופע של אות ייחוס. המאפיין העיקרי של PLL הוא יכולתו לסנן אותות, שאליהם מתוספים רעשים חזקים, ולעקוב ללא שגיאה אחרי התדירות והמופע שלהם. בעזרתם ניתן לבנות מקלטים בעלי רגישות הרבה יותר טובה בהשוואה למקלטים הרגילים. בעזרת PLL אנו נועלים מתנד במקום אחד במערכת, לאותו תדר של מתנד במקום אחר. הוא מאפשר יצירת מתנדים/שעונים עם רעשי פאזה קטנים ביותר . בנוסף הוא מאפשר שימוש במתנד אחד, ולהמיר אותו להרבה מקורות בעלי תדרים שונים.

סכמת מעגל PLL -



איור 12 - מבנה פנימי של מעגל PLL

כאשר :

PD - גלאי מופע המפיק אות שגיאה יחסי להפרש המופעים של אותות המבוא.

LPF - מסנן המפיק מתח בקרה ישר DC ל-VCO. זהו מסנן שמעביר תדרים נמוכים.

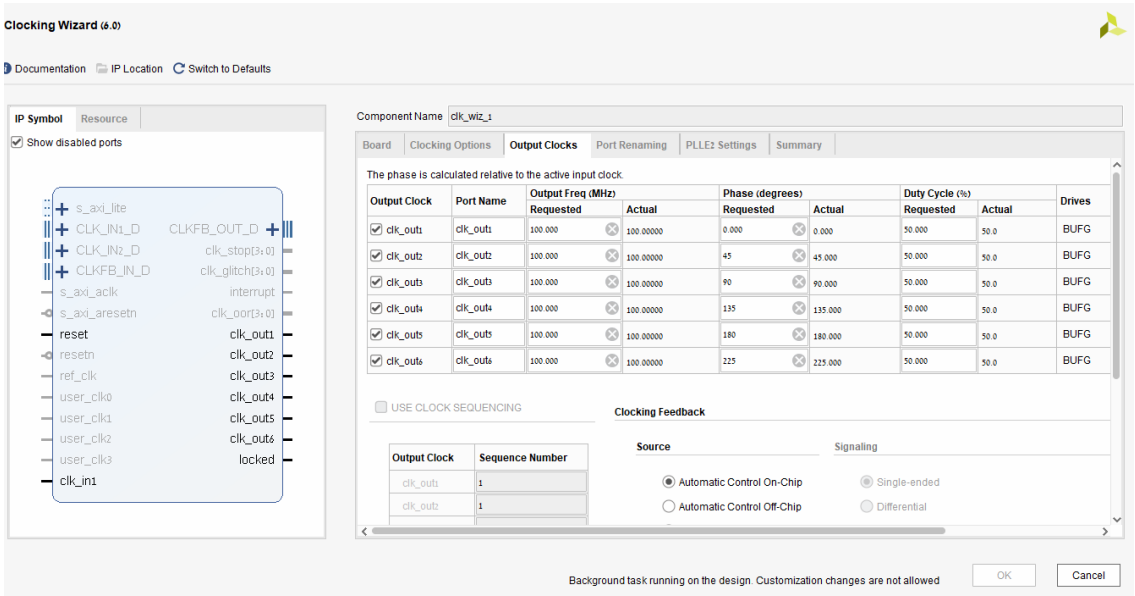
VCO - מתנד מבוקר מתח המפיק אות מחזורי עם תדר היחסי למתח במבוא.

אופן פעולת ה PLL –

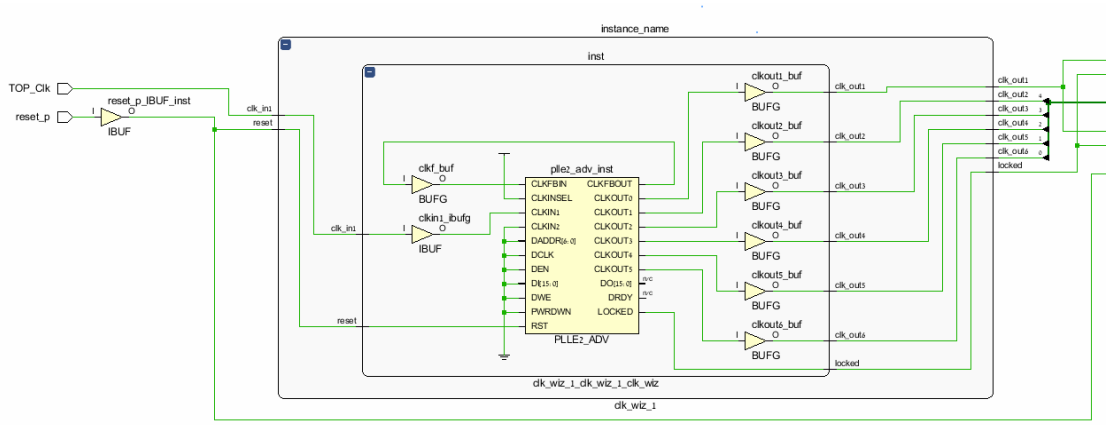
גל נכנס לתוך רכיב ה-XOR, המשמש כגלאי פאזה. כידוע , מוצא ה- XOR הוא 1 לוגי אם הכניסות שונות ו- 0 לוגי אם שוות. תדר שונה לגמרי יגרור מוצא זהה ל- Vcc (1 לוגי) ולהיפך. בתחומי הביניים נקבל רוחבי פולסים משתנים. דרגת ה-LPF ממצעת את השינוי בגל הריבועי היוצא מה-XOR לכדי מתח DC עם אדווה קטנה. ככל שקבוע הזמן של המסנן יהיה גדול יותר, כך האדווה תהיה קטנה יותר, כלומר הדבר ישפר את פעילות ה-PLL ויגרום לתחום עקיבה ונעילה גדולים יותר. דרגת הVCO - בעת שינוי מתח הבקרה ברגל 5 נקבל שוני במתח אליו מגיע הקבל לפני הפיכת המצב בFF - כך משתנה התדר.
ליצירת שעוני העבודה המשתנים בחרנו להשתמש ברכיב הPLL . הרכיב שבנינו נועד ליצירת שעונים שונים בכדי למנוע התמודדות עם בעיות טיימינג עתידיות . ישנם כמה דרכים להבדלת השעונים השונים , אך אנו בחרנו להשתמש בשיטת הזזת הפאזה , כדי ליצור את השעונים הנוספים .

אנו נעבוד עם 7 שעונים :

שעון 1 בעל פאזה 0 אשר נכנס ישירות אל תוך רכיב הLFSR ,
ועוד 6 שעונים בעלי פאזות הולכות וגדלות (45 , 90 , 135 , וכו') אשר בסופו של דבר אחד מהם בכל איטרציה יימצא בכניסת הAES באופן רנדומאלי, דרך רכיב ה LSFR Select וזאת לשיפור מקדם האבטחה ולהעלאת סיבוכיות לשם זליגת המפתח הסודי.



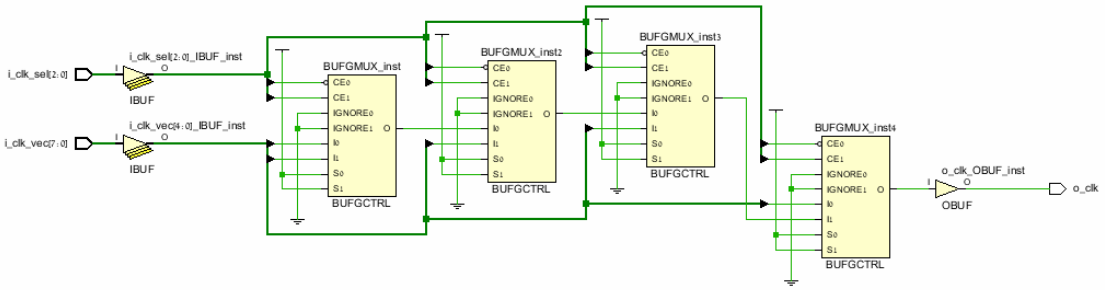
איור 16 – רכיב הPLL בVivado והשעונים המיוצרים בפאזות השונות



איור 17 - פיצול השעונים של הPLL בתוכנת הVivado

- Buffer Mux Global

לאחר יצירת 6 השעונים אשר ישמשו אותנו ליצירת הRandom-Clock , אנו נרצה כל פעם לעבוד עם שעון אחר . לשם הריבוב ביניהם , נרצה להשתמש Mux-Buffer גלובלי אשר מובנה בסביבת העבודה VIVADO. Mux-Buffer גלובלי הינו רכיב מיוחד שנועד להתמודד מול בעיות טיימינג אשר עתידות להיווצר משימוש מרובה שעונים .
לאחר צריבה התברר לנו כי המערכת עובדת בצורה יותר יעילה כאשר אנו קובעים בעצמינו קובעים את מיקומי הBuffer-ים .



איור 18 - Mux-Buffer-Global

LFSR Select

Linear Feedback Shift Register – LFSR

הLFSR הינו מעין רגיסטר מיוחד שבו יחס ההתרחשויות ההולכות ונשנות תלוי במצבו הקודם .

למעשה , הLFSR הינו shift-register שסיבית הקלט שלו בינה פונקציה לינארית של מצבו הקודם . הפונקציה הלינארית הנפוצה ביותר של ביטים בודדים היא בדרך כלל תהיה פונקציית XOR .

אופן הפעולה של הLFSR –

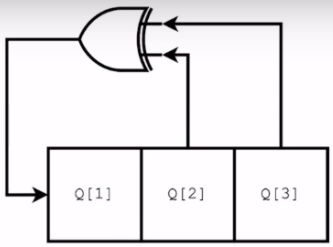
כמו שראינו לעיל , הLFSR בנוי ממספר סופי של ביטים בודדים אשר נכנסים לשער XOR ותוצאתם משורשרת אל תוך הMSB של המערך אשר עתיד להיכנס שוב לתוך השער .
בצורה הזו ,ובעזרת היותו Shift Register אנו יוצרים מעין חילוף מהיר של הביטים הבודדים במערך ובעצם משנים את ערכו בכל איטרציה בצורה יחסית מהירה ומחזורית .

לדוגמא :
אם נכניס את המספר 1 בצורה בינארית או למעשה את הרצף :
$Q(1) = 0$, $Q(2) = 0$, $Q(3) = 1$ ונקבל במוצא 1 מאחר ו$Q(2) \neq Q(3)$. אותו 1 יחלחל אל תוך $Q'(1)$ ואילו $Q'(2)$ יקבל את $Q(1)$ הישן (0) ו$Q(3)$ יקבל את $Q'(2)$ הישן (0) .

כעת קיבלנו רצף חדש $Q'^{(1)}Q'^{(2)}Q'^{(3)} = 100 = 4$

כאשר הביטים $Q'(2)Q'(3)$ יכנסו חזרה אל שער ה*XOR* , נקבל 0 במוצא השער שיכנס ישירות את תוך $Q'(1)$. תתבצע הזזה שמאלה ונקבל את הרצף $Q'^{(1)}Q'^{(2)}Q'^{(3)} = 010 = 2$.

ומאחר ואנו משתמשים ב3 ביטים , נוכל לקבל 7 מספרים שונים במערך ה*LFSR* , ו לאחר מספר איטרציות ,אנו נחזור חזרה למספר 1 ונתחיל את כל התהליך מחדש באופן מחזורי .



1	0	0	1
4	1	0	0
2	0	1	0
5	1	0	1
6	1	1	0
7	1	1	1
3	0	1	1
1	0	0	1
4	1	0	0

איור 19 - מבנה פנימי של LFSR בעל 3 ביטים

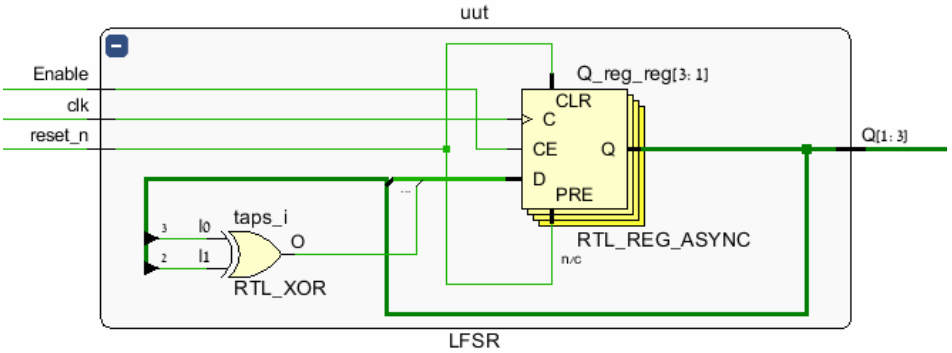
לLFSR ישנם 2 תפקידים עיקריים הפרויקט שלנו .

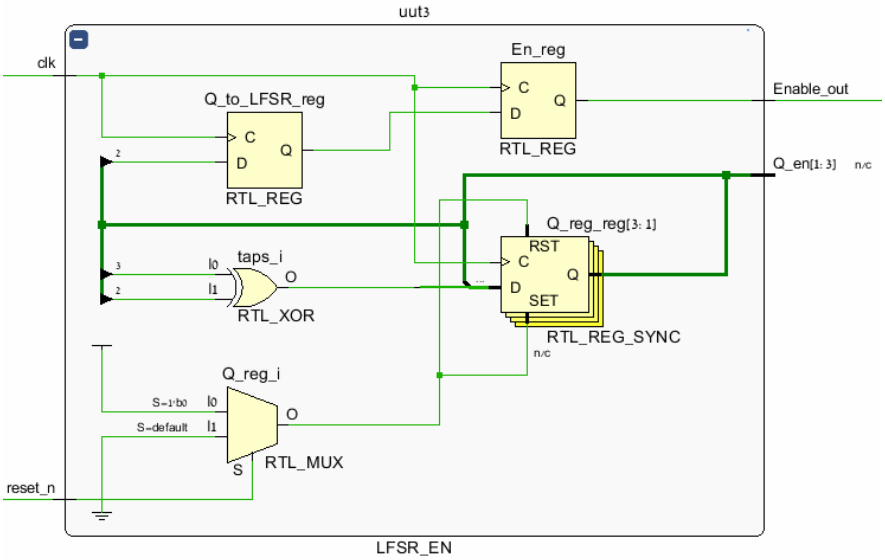
התפקיד הראשון הוא בכדי לשמש ENABLE לLFSR השני בכדי שהLFSR השני יוכל לעבוד עם רכיב הPLL בצורה טובה יותר (לאחר שהתחלנו לעבור על הRandomClock , הבנו שנצטרך להוסיף את רכיב הLFSR_Enable). בצורה הזו , אנו מסוגלים "להוריד" את התדר ולאפשר לקחת מתוך רכיב הPLL את השעון בעל פאזה 0 ובכך נעבוד עם אותו השעון ולא נצטרך ליצור למענו שעון אחר בשבילו. כך נוכל להימנע לבעיית טיימינג שעתידות להופיע .
בנוסף, כך מוצא ה LSFR משמש כמעין רגל SELECT אל רכיב ה Mux-Buffer הגלובלי בכדי שהוא יוכל לעבוד עם מספר שעונים שונים , תוך סיכון נמוך מול בעיות הטיימינג.

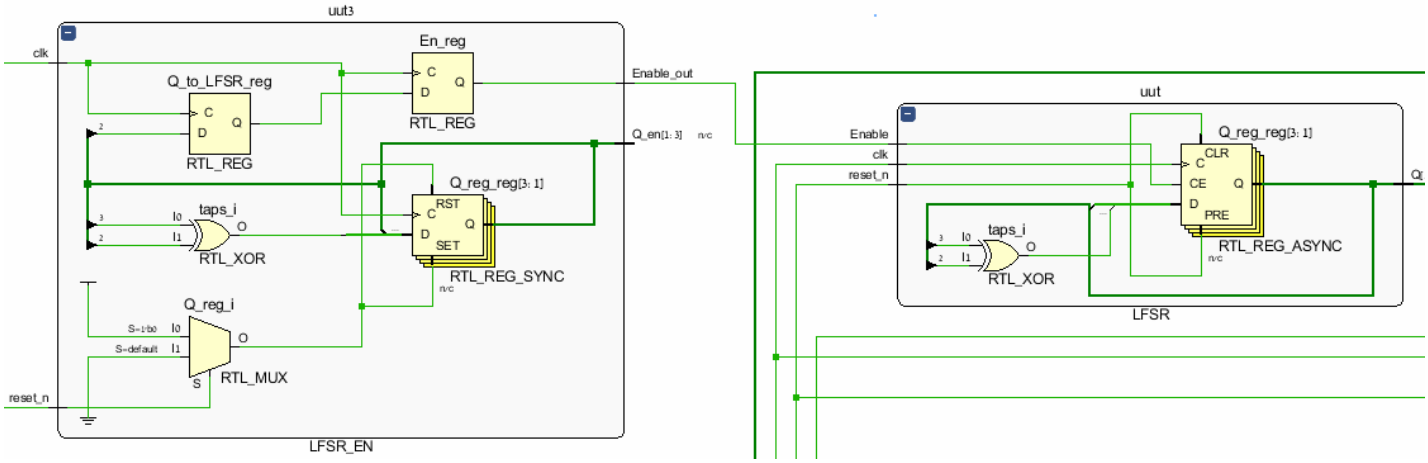
LSFR Enable

רכיב זה זהה לגמרי לרכיב LSFR הרגיל בתכונותיו ובאופן פעולתו . במקרה שלנו , בחרנו להוסיף רכיב LSFR נוסף הנקרא LSFR Enable שלמעשה יהווה כמעין רגל Enable ל LSFR Select . רכיב זה עובד כאשר במוצא קיים מספר אחד מתוך ה7 האפשריים (במקרה של 3 ביט) אשר קבוע מראש (לדוגמא 1). בצורה הזו הLSFR Enable_ מאפשר ל LSFR Select לעבוד רק חלק מוקצב כרצוננו מזמן העבודה אשר וידוע מראש , כך בפועל תדר השעון של הLFSR_Select נקטן.

מימוש הLFSR וה- LFSR_EN :

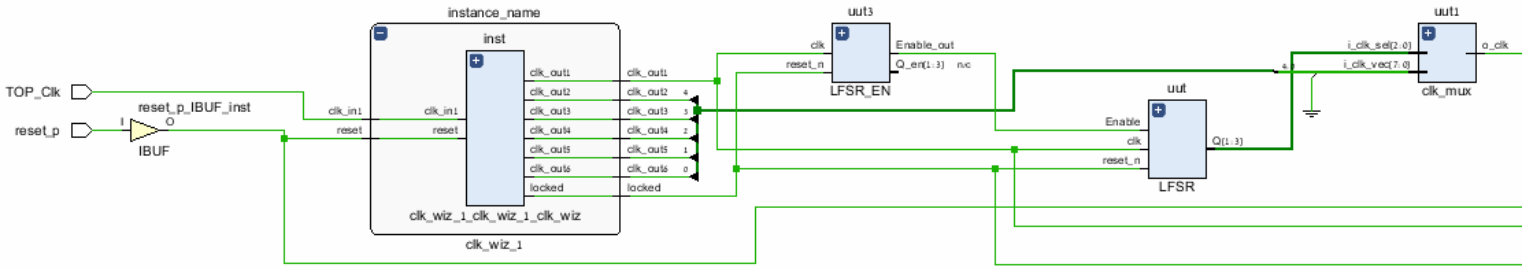




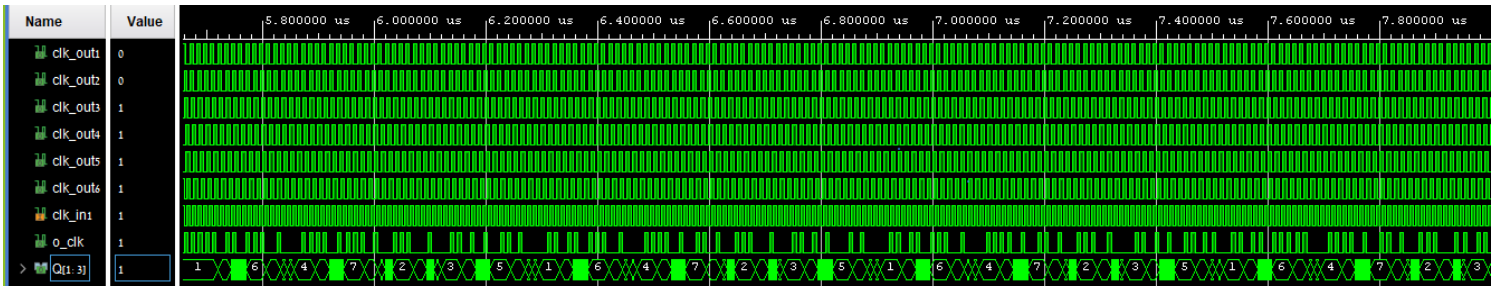


מימוש מערכת הRandom Clock :

בצורת החיבור הזו , אנו מקבלים את הRandom-Clock ו6 שעונים בעלי פאזה שונה כל אחד מהשאר . בעזרת ה Mux-Buffer הגלובלי וה LSFR Select , נקבל כל פעם שעון שונה במוצא הRandom-Clock אשר יכנס אל תוך מערך הAES שלנו .



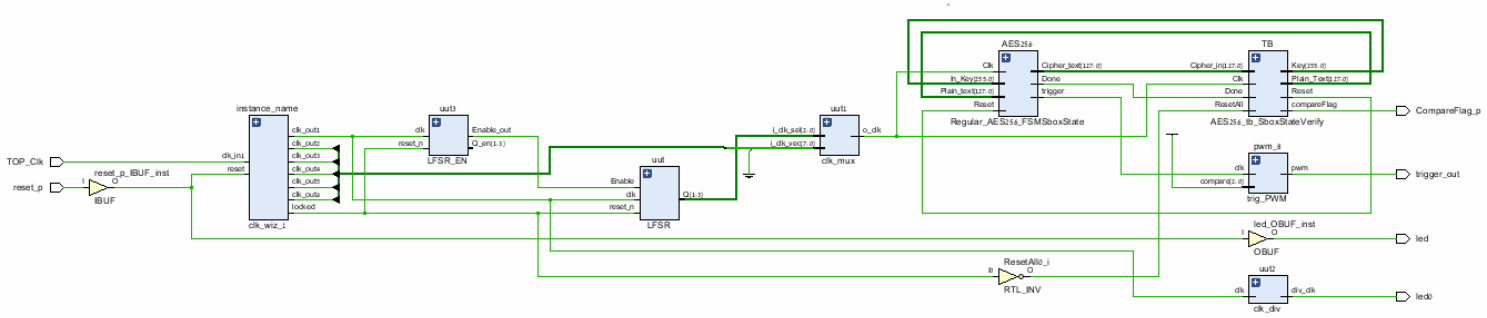
Random Clock System - 21 איור



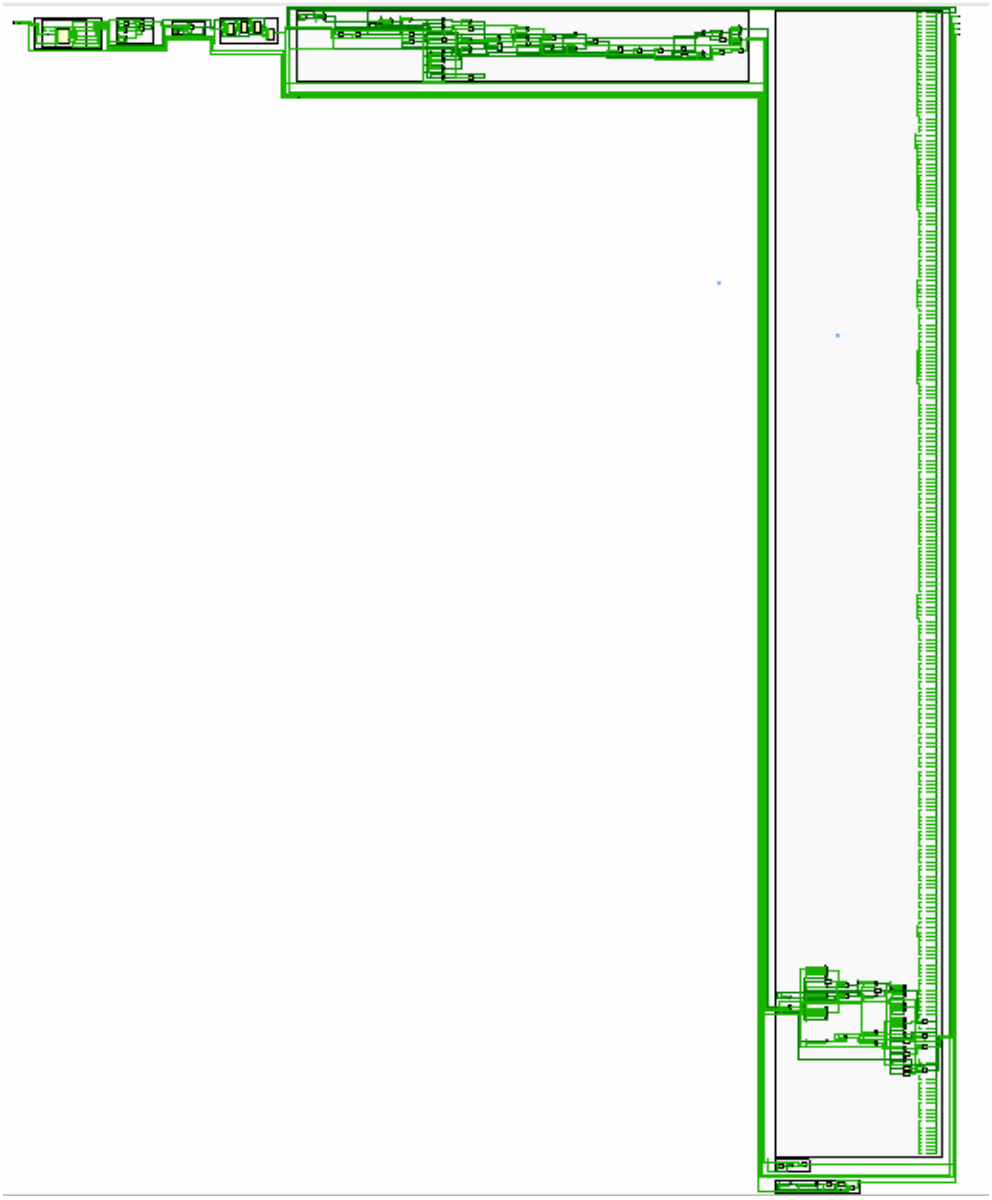
Random Clock Simulation - 22 איור

ניתן לראות מתוך הסימולציה את קבלת הRandomClock כאשר השעון משתנה בקצב פסאודו-אקראי כתלות בQ הנוכחי (LFSR השולט על הSelect של הMux) .

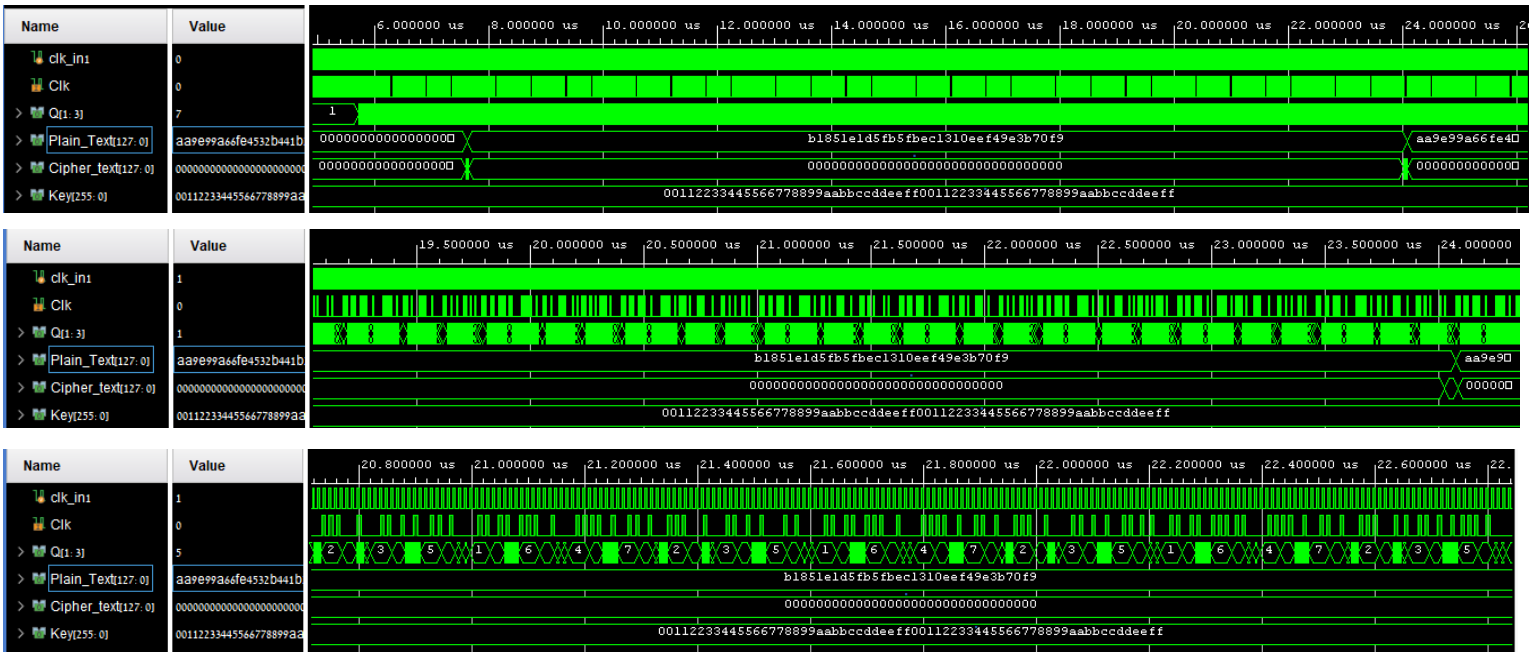
כעת , לאחר הוספת לוגיקת הAES לפרויקט הRandom-Clock , נבדוק שהצפנת הAES החדש , או AES Random-Clock , עדיין נשמרת . נחווט בין לוגיקת הAES עם לוגיקת הRandom-Clock וניצור את הAES Random-Clock , כך שהשעון במוצא ה-Random Clock יהיה השעון בכניסת הAES , ורגל ה Reset תחווט עם רגל הLock של רכיב ה-PLL .



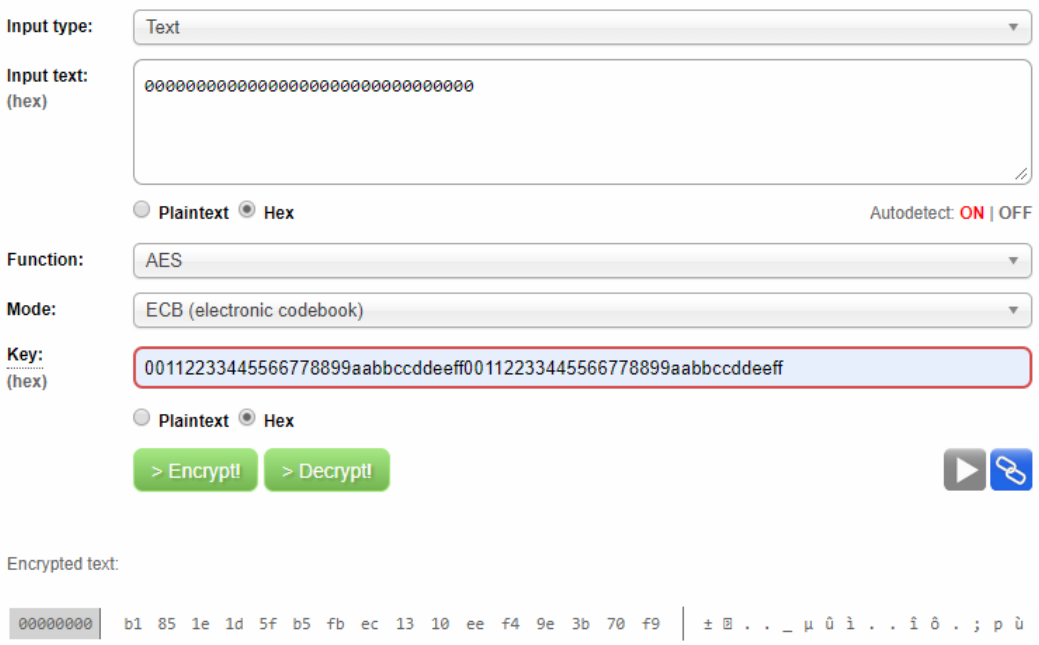
איור 23 - תרשים זרימת המידע של מערכת הRandomClock AES



איור 24 - תרשים מוגדל של זרימת המידע של מערכת הRandomClock AES



איור 25 - תוצאות סימולציית ה AES Random Clock System (ברמת zoom שונה)



איור 26 - תוצאות מחשבון הAES

ניתן לראות לפי השוואה , עם מחשבון הAES , שגם לאחר הוספת הRandom-Clock , תוצאת ההצפנה זהה לחלוטין לחישוב המערכת .

מערך התקיפה:

לאחר שסיימנו ליצור את מערך ההגנה שלנו , נרצה לבדוק את יעילות האבטחה . לשם כך נרצה לבנות מעין מערך התקפה עצמית כלפי מערך ההגנה בכדי לוודא שההצפנה נשמרת ואף השתפרה .

מערך ההתקפה יצטרך לכלול מספר עקרונות בסיסיים :

1. נבצע מספר רב של הצפנות (מקסימום 65,280) על גבי כרטיס הפיתוח בעל מפתח ההצפנה לגילוי כניסות.
2. נבחר נקודה ברצף ההצפנה , ניחוש , בשביל הבייט (*byte*) הראשון של המפתח בסבב ההצפנה הראשון.
3. מציאת מקדמי הקורלציה בין הנתונים הנמדדים לבין הניחוש של המפתח.
4. בחירת ההיפותזה בעלת הקורלציה הגבוהה ביותר שהיא בעצם הסיכוי לניחוש הנכון ביותר. נחזור על סעיפים 4 ו-3 עבור כל הבייטים , עד לקבלת המפתח המלא .

לשם מימוש מערך התקיפה ייצרנו תוכנה לעיבוד הנתונים בעזרת תוכנת ה- *MATLAB*. קוד המימוש מצורף בנספחים .

כעת ,נרצה לפרט על מערך התקיפה ועל האלגוריתם הממומש לבחינת מערך ההגנה ויעילותו:

1. ראשית , נגדיר את הפרמטרים השונים לעבודה כמו גודל המידע המוצפן , גודל המפתח להצפנה, מספר מדידות לביצוע, מספר הנקודות הנמדדות בכל בדיקה, ועוד.
2. שנית , נסרוק ונקרא את מגוון הנתונים המתקבלים במדידות השונות , אותם נכניס אל תוך מטריצה הנקראת מטריצת *P* , המופיעה בנוסחה 7 . המטריצה בנויה כך שמספר העמודות זהה למספר מדידות ההספק שבוצעו בכל הצפנה , ומספר השורות זהה למספר ההצפנות הכולל שבוצעו .

$$\mathbf{P} = \overrightarrow{\text{time samples}} \begin{pmatrix} p_{0,0} & \cdots & p_{0,T-1} \\ p_{1,0} & \cdots & p_{1,T-1} \\ \vdots & \ddots & \vdots \\ p_{n-1,0} & \cdots & p_{n-1,T-1} \end{pmatrix} \Big\downarrow \text{traces}$$

נוסחה 7 – מטריצת נתוני מדידות ההספק

3. בשלב השלישי , נקרא את המידע הכולל (*Plain Text*) המשתנה במהלך המדידות , אותו נכניס אל תוך שורות המטריצה כך שתתקבל מטריצה *X* . מטריצה זו בנויה כך שמס העמודות שלה זהה למספר הביטים הקיימים במידע הכולל (*Plain Text*) ומספר השורות זהה למספר ההצפנות אשר בוצעו בזמן התקיפה .
4. בשלב ברביעי המיקוד העיקרי יכוון לביצוע ההיפותזה על פי נקודה קבועה באלגוריתם ההצפנה . נראה מהם השלבים לבניית ההיפותזה :
   א. מיד לאחר פעולת ההחלפה הראשונה של קופסת ה*S* (*S-Box*) , נבחר ונשתמש בנקודה בסבב ההצפנה הראשון .
   ב. תוך מעבר על כלל ההיפותזות האפשריות עבורנו (0-255) ננסה לגלות מקטעים של *BYTE* אחד אחרי השני בכדי לנחש את הבייט הנכון ביותר מהמפתח .

ג. נשתמש בפעולת *XOR* על כל ניחושי המפתח האפשריים אל מול כל עמודה במטריצה *X* , כך שכל עמודה מייצגת *BYTE* מהמידע המוצפן.
את התוצאות נכניס למטריצה חדשה אשר נקראת מטריצה *XxorK* , בעלת מספר שורות ככמות הצפנת המדידות שביצענו, אל מול 256 שהם כל האופציות האפשריות עבור ניחוש מוצלח של *BYTE* אחד בודד .

ד. המטריצה *XxorK* המתקבלת , תעבור דרך קופסת ה*S* (*S-box*) ,כך שמתקבלת מטריצה חדשה *B* אשר זהה בממדיה למטריצה *XxorK*.

$$\mathbf{B} = \begin{array}{c} \xrightarrow{\text{256 key candidates}} \\ \begin{pmatrix} b_{0,\,0x00} & \cdots & b_{0,\,0xFF} \\ b_{1,\,0x00} & \cdots & b_{1,\,0xFF} \\ \vdots & \ddots & \vdots \\ b_{n-1,\,0x00} & \cdots & b_{n-1,\,0xFF} \end{pmatrix} \end{array} \Big\downarrow \text{traces}$$

נוסחה 8 – מטריצת B-Sbox(XxorK)

כעת , המטריצה החדשה המתקבלת *B* מכילה את כלל הניחושים האפשריים עבור ה*BYTE* הרלוונטי של המפתח , ומטריצה זו למעשה תהווה בסיס בחיפוש ומציאת הקורלציה בין המדידות לניחוש המפתח.

בשלב החמישי , לפני השלב האחרון , בכדי למצוא את הקשר בין הניחושים למדידות, נתחיל בפעולות למציאת קורלציה על פי עקרון *CPA*, *Correlation Power Analysis*, כך שבעצם נחפש קורלציה (התאמה) בין מטריצת *P* למטריצות מודל ההספק שנציג כעת .

מודלי ההספק שבעזרתם ננתח את ההיפותזות מוצאות באופן הבא :

מודל ה *HW* – מודל הספק זה מתבסס על משקל המספר הבינארי המתקבל במוצא קופסת ה*S* – (*S-Box*) , אשר יכמת בכל מוצא בינארי את מספר הביטים ה"דולקים" (כלומר מספר ה-'1'-ים) . זאת מתוך ההנחה כי קיימת קורלציה בין כמות האנרגיה הנדרשת לבין מספר הביטים ה"דולקים" . הסכום מוצג בצורה הבאה :

$$HW = \sum_{i=0}^{n} (bit\_out(n) ==' 1') \ , \ \ n = 0,1,2 \ldots$$

נוסחה 9 – חישוב משקל Hamming

מודל *HD* – מודל זה מתבסס על מספר הביטים המשתנים בעת המעבר בקופסת ה-S (S-Box) בניגוד למודל HW . כאן ההנחה היא כי כמות הביטים ששונו היא בעלת קורלציה לכמות האנרגיה הנצרכת , תוך הזנחת הביטים שלא השתנו (נותרו סטטיים) ללא תלות במצבם . סכום זה מוצג בצורה הבאה :

$$HD = \sum_{i=0}^{n} (bit\_out(n) \oplus bit\_in(n) ==' 1') \ , \ \ n = 0,1,2 \ldots$$

נוסחה 10 – חישוב מרחק Hamming

תוצאות המודלים עבור שינויי ביטים :

Bit transition	HW	HD
$0 \rightarrow 0$	0	0
$0 \rightarrow 1$	1	1
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	1	0

טבלה 4 –מודלי הכוח כתלות בשינויי הביטים

ראשית , נרצה לקבל מדד כמותי על כמות הביטים ה"דולקים" בכל ניחוש , מאחר וצריכת ההספק הרגעית משתנה ביחס לכמות הביטים הפעילים. לכן , נפעיל קודם את אלגוריתם *HW* על המטריצה *B*. במוצא האלגוריתם מתקבלת מטריצה *H* , אשר בעלת אותם ממדי גודל למטריצה *B* . בנוסף , נרצה להפעיל גם את אלגוריתם *HD* על מטריצה *B* ובמוצאו נקבל את מטריצה *HD* אשר גם היא זהה בממדיה למטריצה *B* .

$$\mathbf{H} = \mathrm{HW}\,(\mathbf{B}) = \overbrace{\begin{pmatrix} h_{0,\,0\mathrm{x}00} & \cdots & h_{0,\,0\mathrm{x}FF} \\ h_{1,\,0\mathrm{x}00} & \cdots & h_{1,\,0\mathrm{x}FF} \\ \vdots & \ddots & \vdots \\ h_{n-1,\,0\mathrm{x}00} & \cdots & h_{n-1,\,0\mathrm{x}FF} \end{pmatrix}}^{256 \text{ key candidates}} \Bigg\downarrow \text{traces}$$

נוסחה 11 – מטריצת משקלי Hamming

5. בשלב האחרון נרצה להשתמש במטריצות *HD* ו- *HW* וכמן כן במטריצה *P* שיצרנו המכילה את כלל מדידות. כעת , נחפש את הקורלציה בין הניחוש לבין המדידות ,במטרה למצוא את מקדם הקורלציה הגבוה ביותר אשר כנראה יהיה לניחוש הנכון ביותר על *BYTE* מסוים במפתח . את כלל מקדמי הקורלציה נאחד למטריצה אחת הנקראת מטריצת *correlation_HW* כך שכל עמודה תייצג את מקדמי הקורלציה של הניחושים הספציפיים , קרי הניחוש של *BYTE* ספציפי במפתח .

למציאת הקורלציה , נחשב את מקדמי הקורלציה וניעזר בנוסחת מקדם המתאם של Pearson (**Pearson correlation coefficient**) . תוצאות הנוסחה יהיו ערכים בין (1) ל (1-) כאשר:

(1) – מייצג קורלציה מלאה ,

(0) – מייצג כי לא קיימת כלל קורלציה בין הנתונים השונים ,

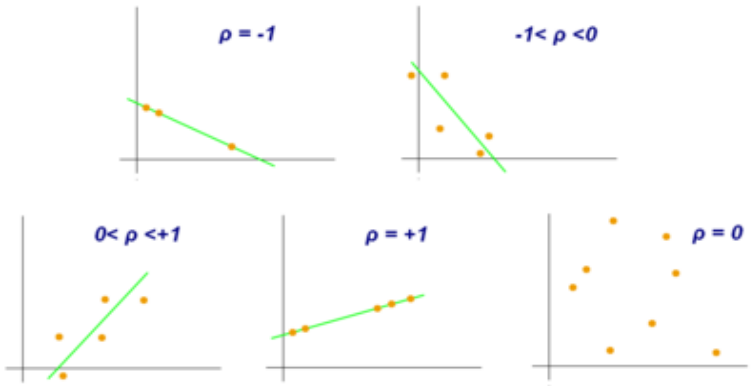
(1-) – מייצג קורלציה מלאה בעל יחס הפוך .

הנוסחה מנוסחת באופן הבא :

$$\rho_{i,j} = \frac{\sum_{k=1}^{n_{trc}} \left[\left(h_{k,i} - h_{i_{avg}} \right) \cdot \left(p_{k,j} - p_{j_{avg}} \right) \right]}{\sqrt{\left[\sum_{k=1}^{n_{trc}} \left(h_{k,i} - h_{i_{avg}} \right)^2 \right] \cdot \left[\sum_{k=1}^{n_{trc}} \left(p_{k,j} - p_{j_{avg}} \right)^2 \right]}}$$

$$i = 1, 2, \dots, 256 \quad , \quad j = 1, 2, \dots, read\_trc$$

נוסחה 12 –מציאת מקדמי הקורלציה באמצעות מקדם המתאם של Pearson



איור 27 – תרשימי פיזור מתאם Pearson

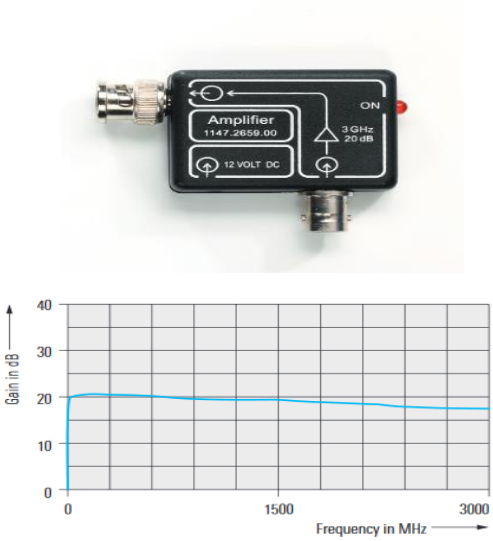
את איכות הגילויים של האלגוריתם ניתן להעריך לפי מדד ה-*CR* (*Correlation Ratio*) אשר למעשה הוא היחס בין הקורלציה הטובה ביותר שנמצאה לבין הקורלציה השנייה הטובה ביותר. איכות הגילוי תלויה בגובה כמות שמדידות אשר בוצעו ועליהם הופעל האלגוריתם . כלומר , עם כמות גבוהה יותר של מדידות נוכל לאמוד בצורה טובה יותר את איכות הגילוי. נרצה שמדד ה-*CR* יהיה גדול כמה שניתן (לפחות *CR>1*) ויישאר מעל הערך 1 , כך שהניחוש הנבחר יהיה בעל סיכוי גדול יותר להיות נכון .

התוקף ינסה לשמור על מספר מדידות נמוך , מאחר וכמות גדולה יותר של מדידות מאריכה את משך זמן התקיפה הכולל אשר נדרש על הרכיב .

מדידות :

בכדי למדוד את תוצרי הפרויקט שלנו , נבצע מדידת נתונים באמצעות תקיפה חיצונית יזומה על גבי המערכת המוגנת(AESRandomClock) , אותה נשווה אל מול מדידת הנתונים על גבי המערכת הלא מוגנת(RegularAES) , לשם השוואה והסקת מסקנות .

ציוד המדידה – למדידת האותות השתמשנו במעבדה באוניברסיטת בר אילן בקדם מגבר R&S® and HZ-16 preamplifier set אשר מחוברים לScope ומופיעים באיור 28 :



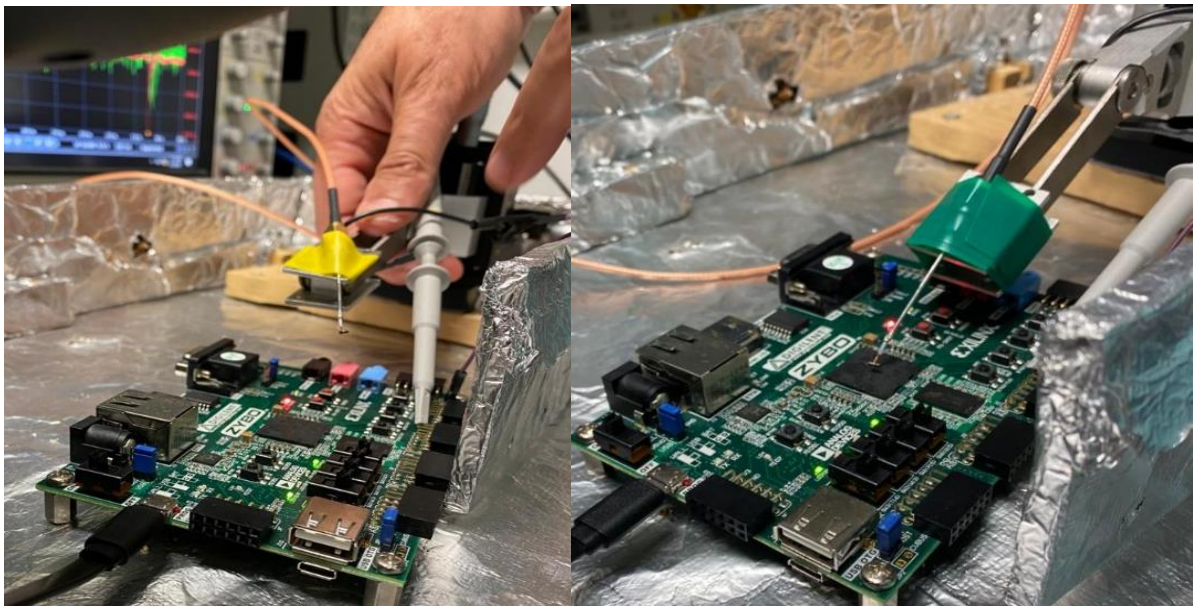
איור 28 – ציוד המדידה

נתבונן בדפי הנתונים של מכשירי המדידה וניראה כי היענות התדר של מכשור המדידה המדובר הינה מיטבית לצרכים שלנו עקב הניחות הגבוה בתדרים נמוכים מ $3\ [MHz]$. בצורה הזו מתבצע סינון של מרבית רעשי הסביבה והרשת אשר פועלים בתדרים נמוכים בהרבה.

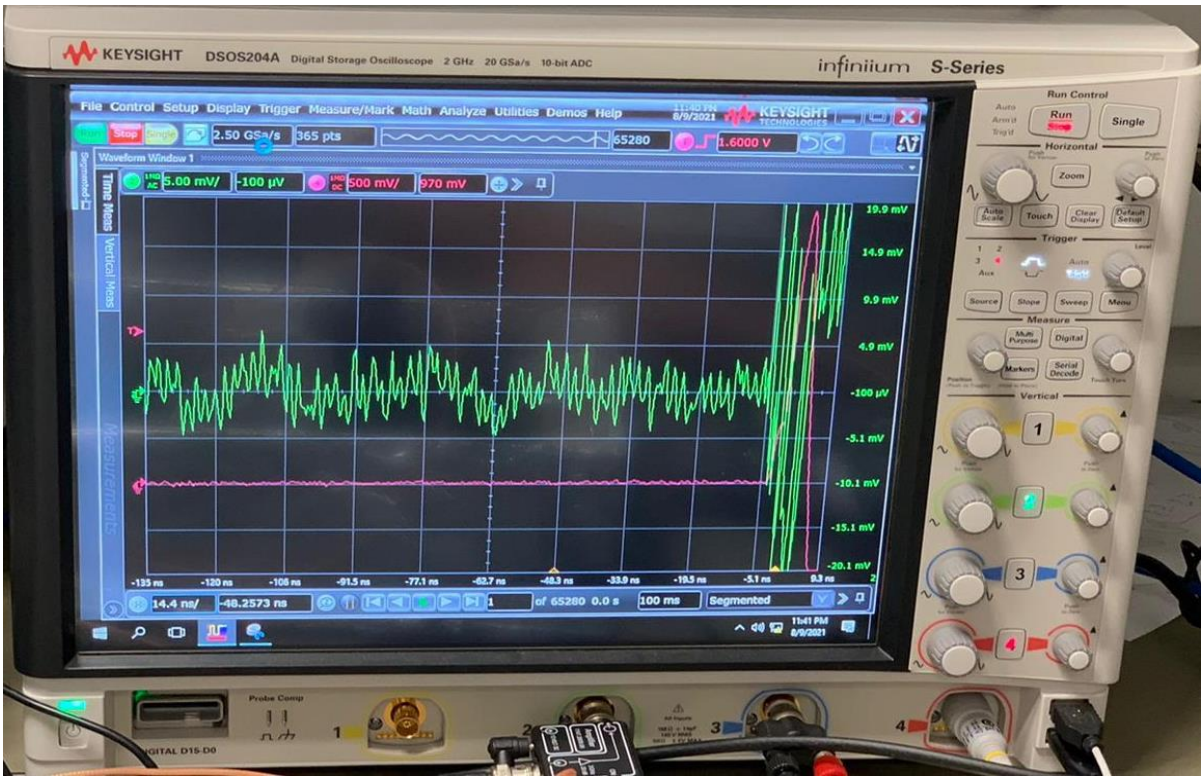
את המדידות ביצענו במעבדתו של ד"ר ויצמן יואב , באוניברסיטת בר אילן . המדידות נעשו באמצעות SCOPE של חברת KEYSIGHT (ראה איור 30) ובעל יכולת דגימה מקסימלית של כ65,280 הצפנות , אשר חובר לסט פרוב וקדם מגבר (ראה באיור 29,28) .ציוד זה מודד את הנתונים (כמתואר באיור 29) על גבי הערכה הצרובה בקוד מערך ההגנה שלנו .הערכה מבצעת 65,280 הצפנות בעלות אותו מפתח בplaintext-ים שונים (המידע המוצפן של הסבב הקודם), ובשביל לאמת את קיום ההצפנה , חישבנו את הצפנה מס 65,280 בעזרת קוד MATLAB, והשוונו את ההצפנה המתקבלת בערכה (קוד הVERILOG) להצפנה המחושבת מראש . כאשר גילינו התאמה , המשמעות היא שסיימנו רצף של 65,280 הצפנות וההצפנה האחרונה תואמת לערך המחושב , הרי שהצלחנו להצפין ונדלק לד חיווי מתאים בערכה .

בעזרת הפרוב המחובר לסקופ (איור 29) , נרצה למדוד את הערכים הנפלטים מהערכה בכדי לייצר את מטריצת P , לכן נדליק את רגל הTrigger בסיומו של הסבב הראשון , ומיד בתחילת הסבב השני. פעולה זו נבצע על כל הצפנה , כלומר נשלח מעין FLAG לרגל אליה מחובר הפרוב מהמשקף תנודות לשם "הודעה ללקיחת דגימה" .

באמצעות חזרה על רצף הפעולות הנ"ל , הצלחנו לאסוף 10 קבצי עיבוד לכל דגימה , על מנת להקטין את השפעתו של הרעש בזמני המדידה ולקבלת תוצאות אמינות יותר.



איור 29 – מהלך ואמצעי המדידה



איור 30 – משקף תנודות במהלך המדידות

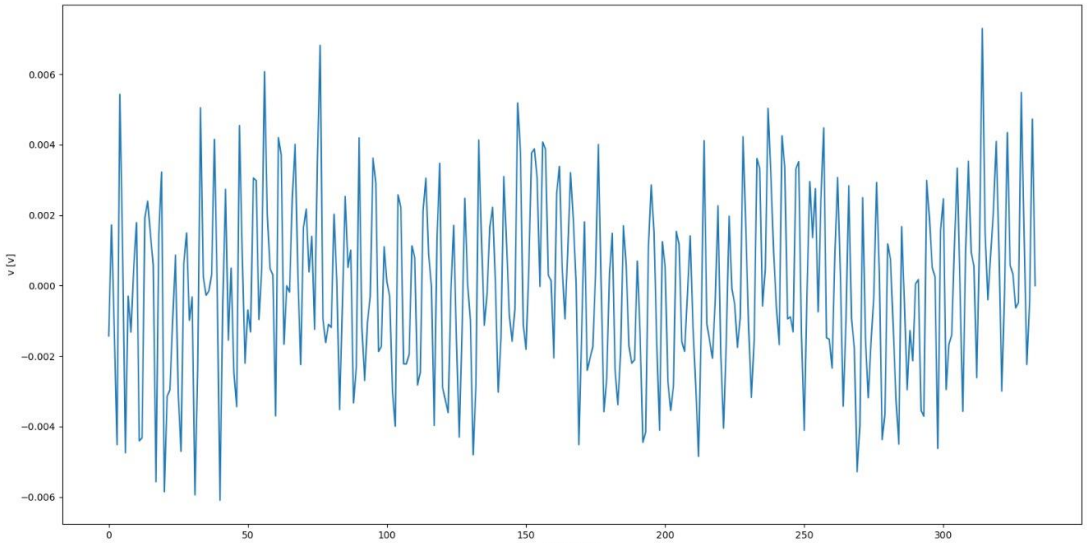
קליטת ועיבוד הנתונים :

את הנתונים הנמדדים שמרנו ב3 סוגי קבצים שונים (.mat , .csv , .txt) בכדי למצוא את דרך קליטת הנתונים האופטימלית . לאחר מכן , את הנתונים קלטנו בעזרת תוכנה בשפת Python וביצענו על כל אחת מהמדידות את הפעולות הבאות :

לכל מדידה הורדנו את תדר הDC בעזרת מעבר דרך מישור התדר ,הורדת תדר ה-0 וחזרה למישור הזמן .

מאחר וניתן לראות מאיור 29 , שהפעלת רגל הTrigger מייצרת רעש גבוה , החלטנו לבצע סינון לפי רמות מתח לקבלת תוצאות נקיות יותר , כך שנותרנו עם 333 דגימות למדידה .

כעת נציג גרף של רצף הדגימות עבור הצפנה אחת בתהליך 65,280 ההצפנות נקי מרעש הטריגר:



גרף 1 –one encryption without trigger noise

לאחר פעולות אלו ביצענו מיצוי לכל סט מדידות , אי לכך נקבל את מטריצת נתוני מדידות ההספק P בגודל 65,280 שורות ו333 עמודות לכל דגימה .

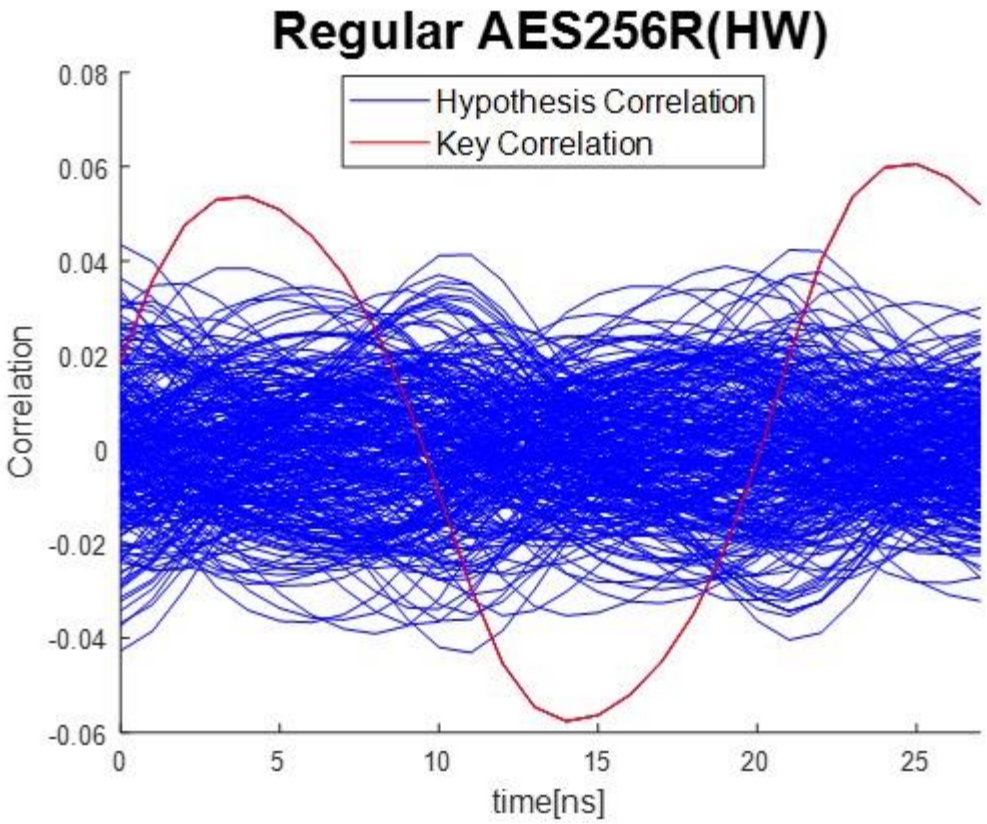
נשמור את מטריצת P בתוכנת הMATLAB לשם המשך תהליך התקיפה .

עיבוד התוצאות:

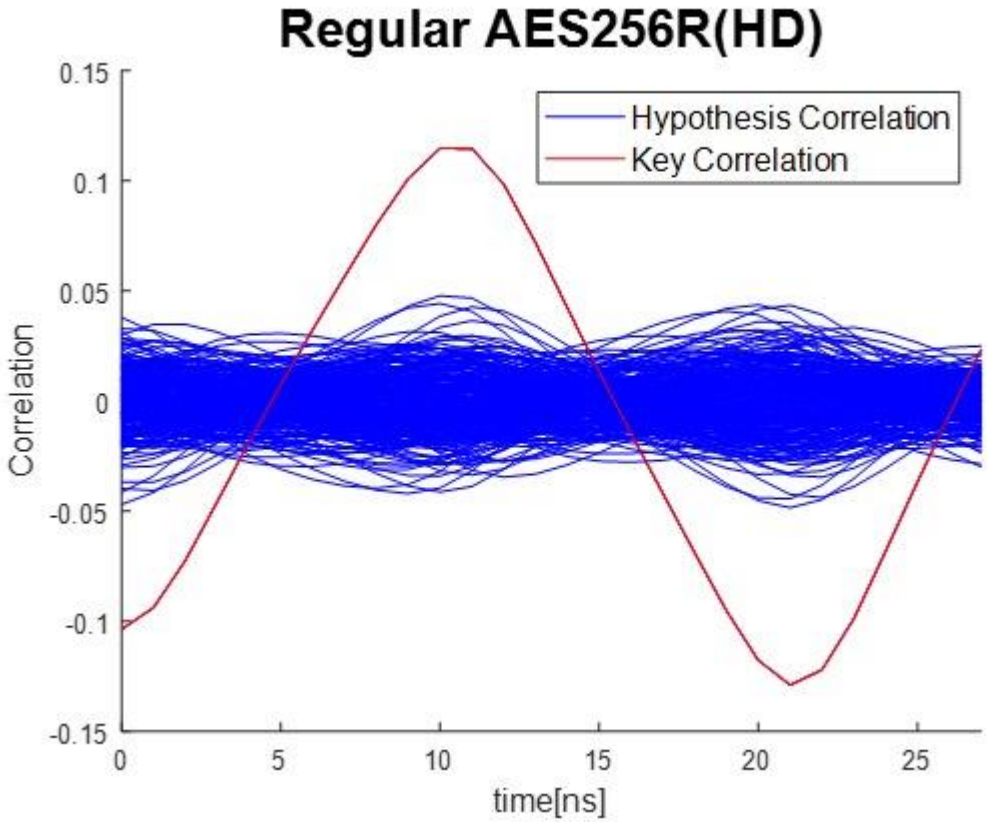
לאחר קליטת המידע מתוכנת הPython ל.MATLAB , ביצענו את קוד אנליזה לאותות הנפלטים בMATLAB כמתואר בפרק מערך התקיפה בספר הפרויקט .

תחילה נרצה לראות כי קוד האנליזה שלנו עובד , כלומר ננסה לפרוץ את המערכת הלא מוגנת שלנו (RegularAES) .

הגרפים הבאים מתארים את הקורלציה המתקבלת בין ההיפותזות לבין מדידות ההספק במישור הזמן למימוש המערכת הלא מוגנת .



גרף 2 -Unprotected Regular AES256 and key correlation over time using HW Hypothesis correlation



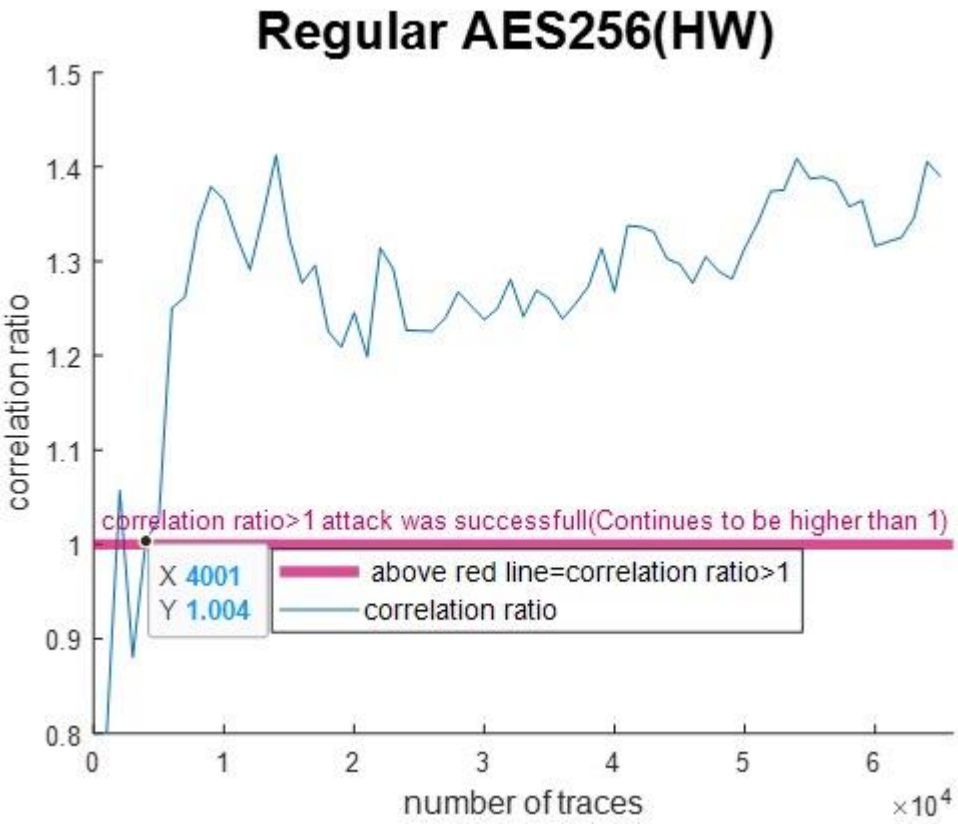
גרף 3 - Unprotected Regular AES256 and key correlation over time using HD-Hypothesis correlation

הגרפים 2,3 מתארים את הקורלציה המקסימלית המתקבלת בין ההיפותזות לבין מדידות ההספק על המערכת Regular AES הלא מוגנת , עבור 65,280 הצפנות .

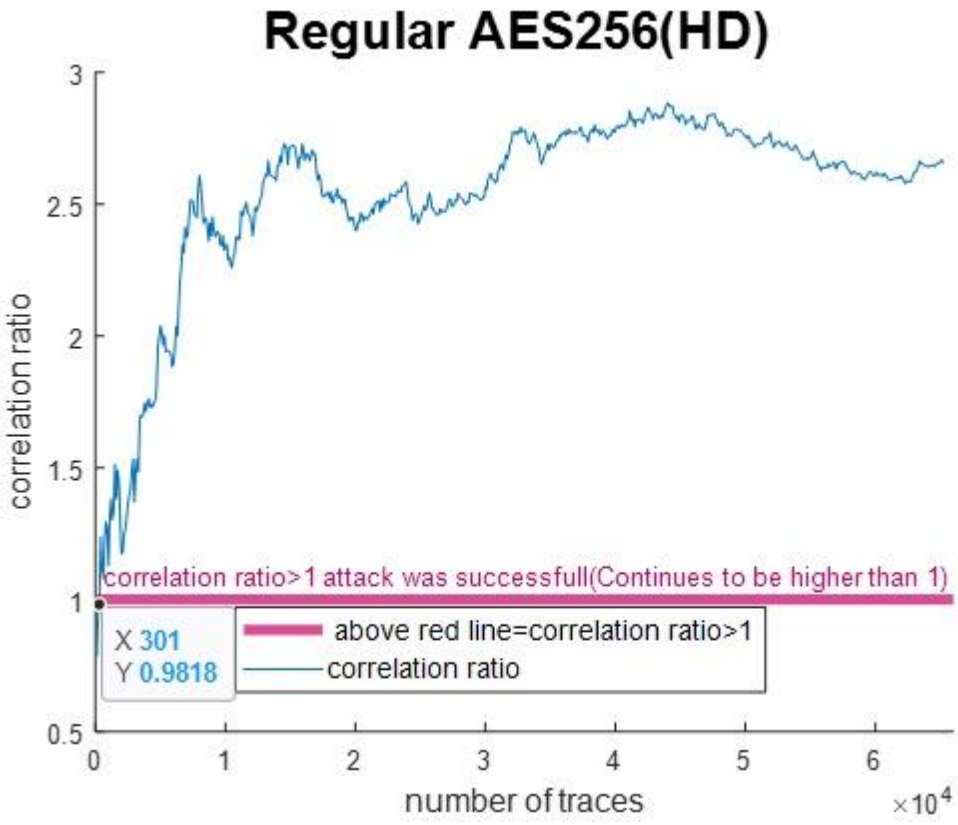
מגרף 2 ניתן לראות כי ביחס לשאר ההיפותזות , הקורלציה של המפתח האמיתי היא המקסימלית. מכאן ניתן להסיק כי ההתקפה הצליחה והחומרה נפרצה בשיטת HW.

גם מגרף 3 ניתן להסיק כי ביחס לשאר ההיפותזות , הקורלציה של המפתח האמיתי היא המקסימלית. מכאן ניתן להסיק כי ההתקפה הצליחה והחומרה נפרצה בשיטת HD .
בנוסף , מגרף 3 לעומת גרף 2 קורלציית המפתח הנכון היא גבוהה יותר ביחס לשאר ההיפותזות האפשריות , מה שמעיד על שיטת HD כיעילה יותר אל מול שיטת HW בתהליך פריצת המערכת הלא מוגנת .

עבור המערכת הלא מוגנת , כעת נציג את מדד הCR (כמתואר בפרק מערך התקיפה) , התלוי בכמות מחזורי ההצפנה הנמדדים .
נזכיר שמדד זה מייצג את טיב הקורלציה קרי , כאשר היא מעל 1 ונשארת מעל ערך זה, קיימת הצלחה בפריצת המערכת.



גרף 4 – Unprotected Regular AES256 HW correlation ratio over number of traces



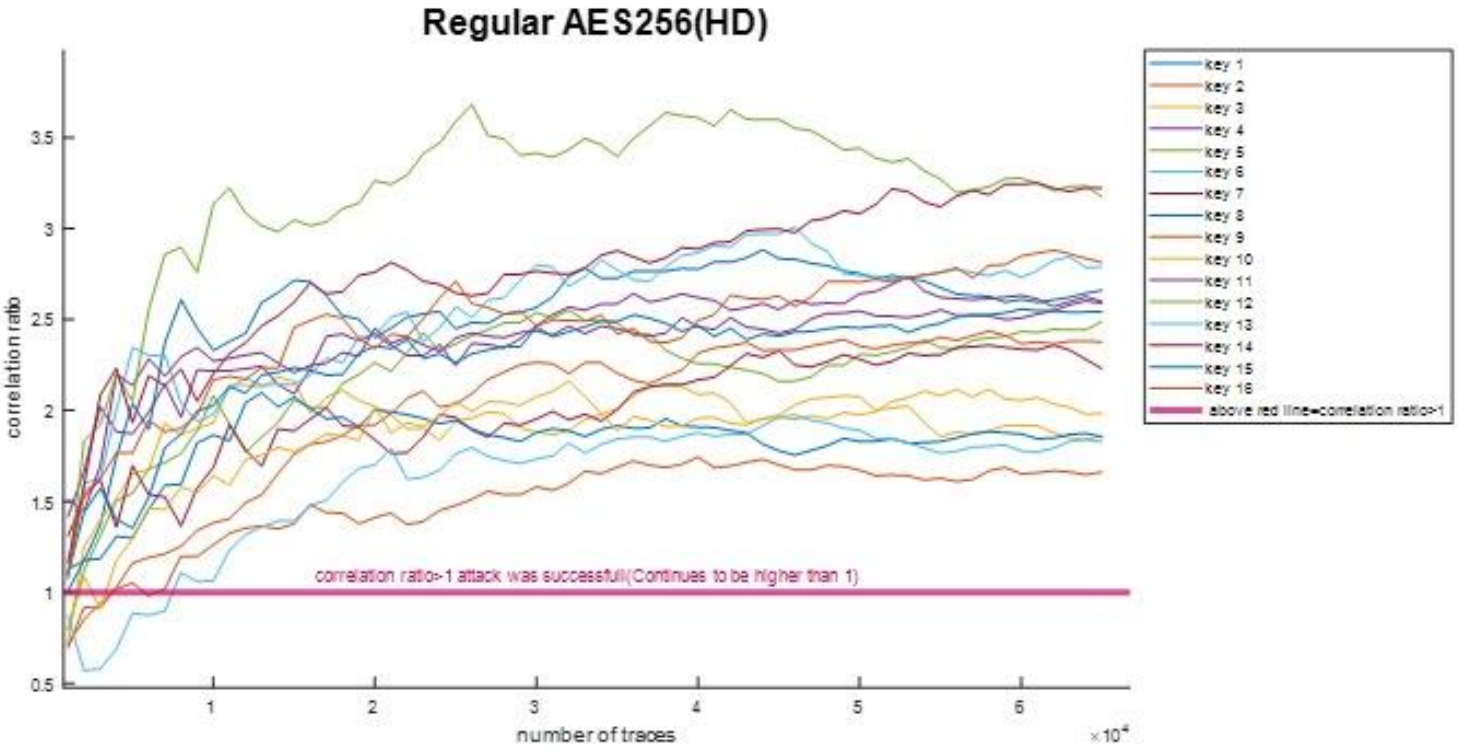
גרף 5 – Unprotected Regular AES256 HD correlation ratio over number of traces

מגרף 4 ניתן לראות כי מדד ה-correlation ratio חוצה את קו ה1 וממשיך לגדול כבר לאחר 4001 הצפנות , לכן אנו יכולים להסיק כי החומרה אינה מוגנת והיא בהחלט פריצה בשיטת HW.

גם מגרף 5 מדד ה-correlation ratio חוצה את קו ה-1 וממשיך לגדול כבר לאחר 301 הצפנות , לכן אנו נסיק החומרה אכן פריצה לחלוטין ואינה מוגנת בשיטת HD.

ניתן גם לראות שבגרף מס 5 לעומת גרף 4החומרה נפרצת בכמות הצפנות קטנה יותר , וגובה ה-correlation ratio גבוה בהרבה . כמו שציינו בפרק מערך התקיפה , ככל שהCR גדול יותר , כך גדל הסיכויים שההיפותזה היא בעלת וודאות גדולה יותר להיות ההיפותזה הנכונה .

עובדה זו מצביעה על כך כי שיטת HD יעילה יותר לתקיפת המערכת הלא מוגנת מאשר שיטת HW . זאת ניתן לראות מגרפים 2-5 ומההסברים הנלווים להם . לכן נתמקד בשיטת HD ונציג גרף CR עבור 16 המפתחות , בכדי לזהות אפשרויות פריצה של המפתחות ע"י התוקף .

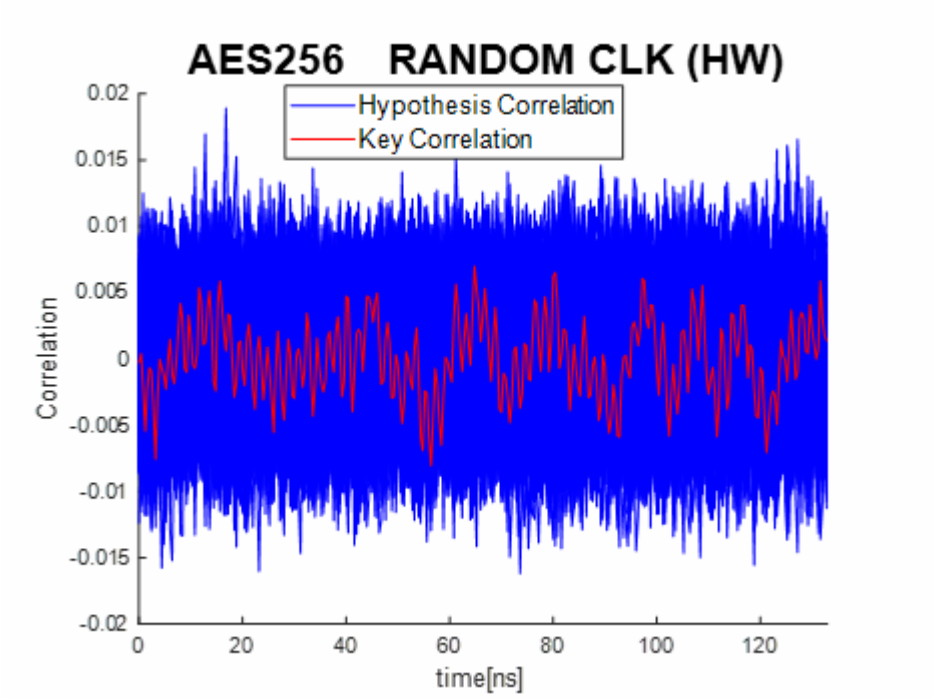


גרף 6 – Unprotected Regular AES256_HD Correlation ratio for 16 keys over num of traces

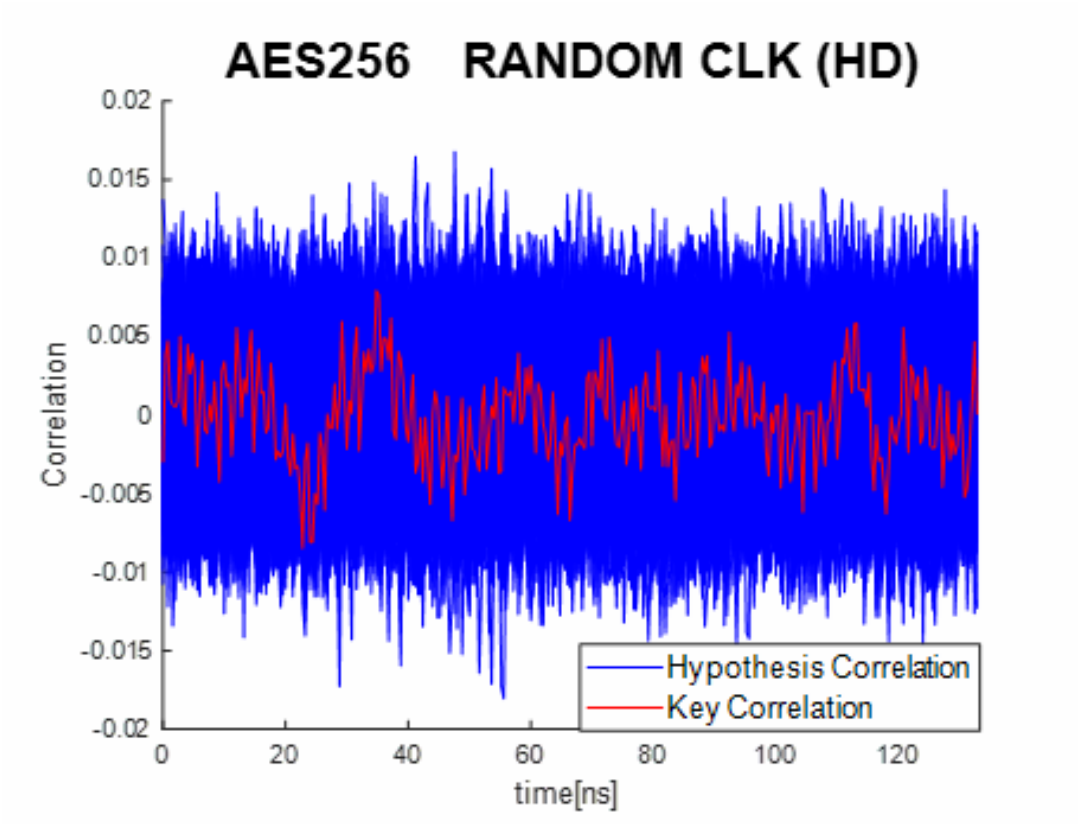
מגרף 6 ניתן להסיק כי מערכת RegularAES הלא מוגנת פריצה בהחלט עבור 16 המפתחות כבר במספר הצפנות נמוך יחסית (מתחת ל10,000 הצפנות) .

כעת , לאחר שראינו כי מערך התקיפה המתוכנן אכן מצליח לפרוץ את המערכת RegularAES הלא מוגנת , נעבור למערכת המוגנת יותר באמצעות טכנולוגיית הRandomClock , וגם אותה נתקוף בעזרת 2 השיטות שהראנו לעיל ונשווה בין 2 המערכות המוגנת והלא מוגנת .
מטרתנו המרכזית היא להראות כי טכנולוגיית הRandomClock אכן משפרת את יכולות ההגנה על המערכת , ולא תיפרץ גם לאחר 65,280 הצפנות .

הגרפים הבאים מתארים את הקורלציה המתקבלת בין ההיפותזות לבין מדידות ההספק במישור הזמן למימוש המערכת מוגנת בעזרת RandomClock .



גרף 7 - RandomClock Protected AES256 Hypothesis and key correlation over time using HW correlation

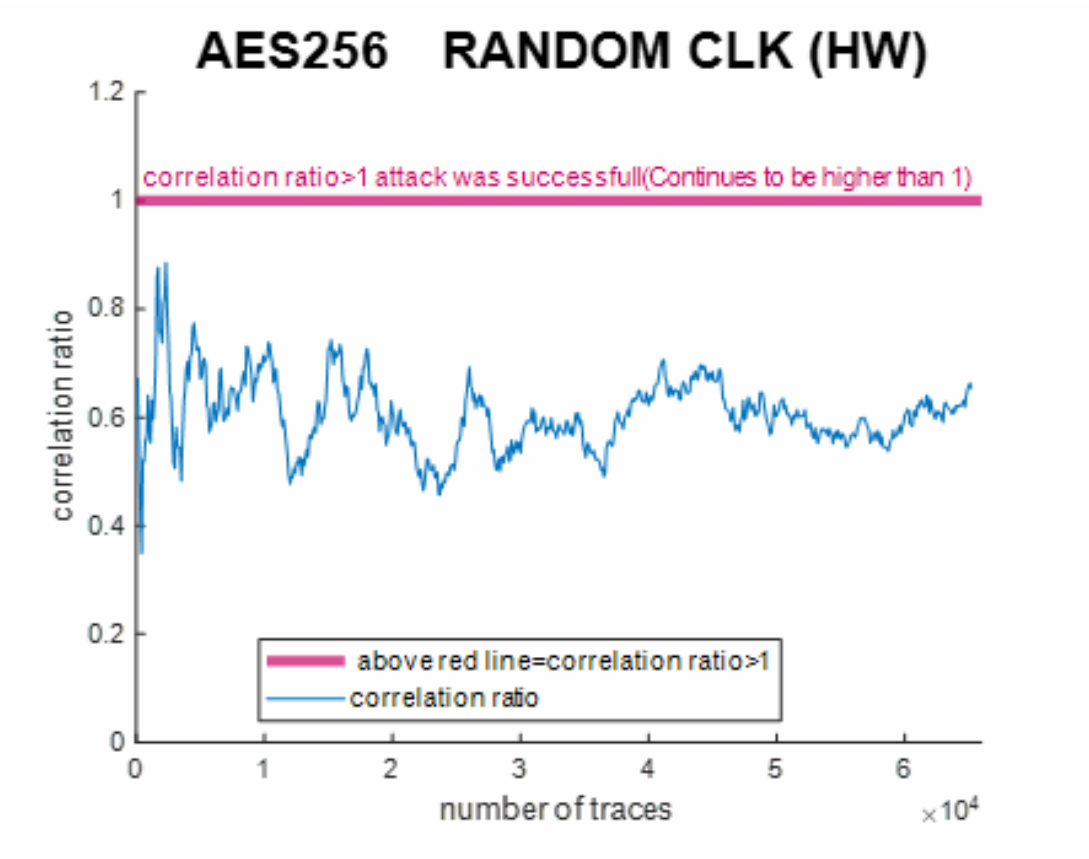


גרף 8 - RandomClock Protected AES256 Hypothesis and key correlation over time using HD correlation

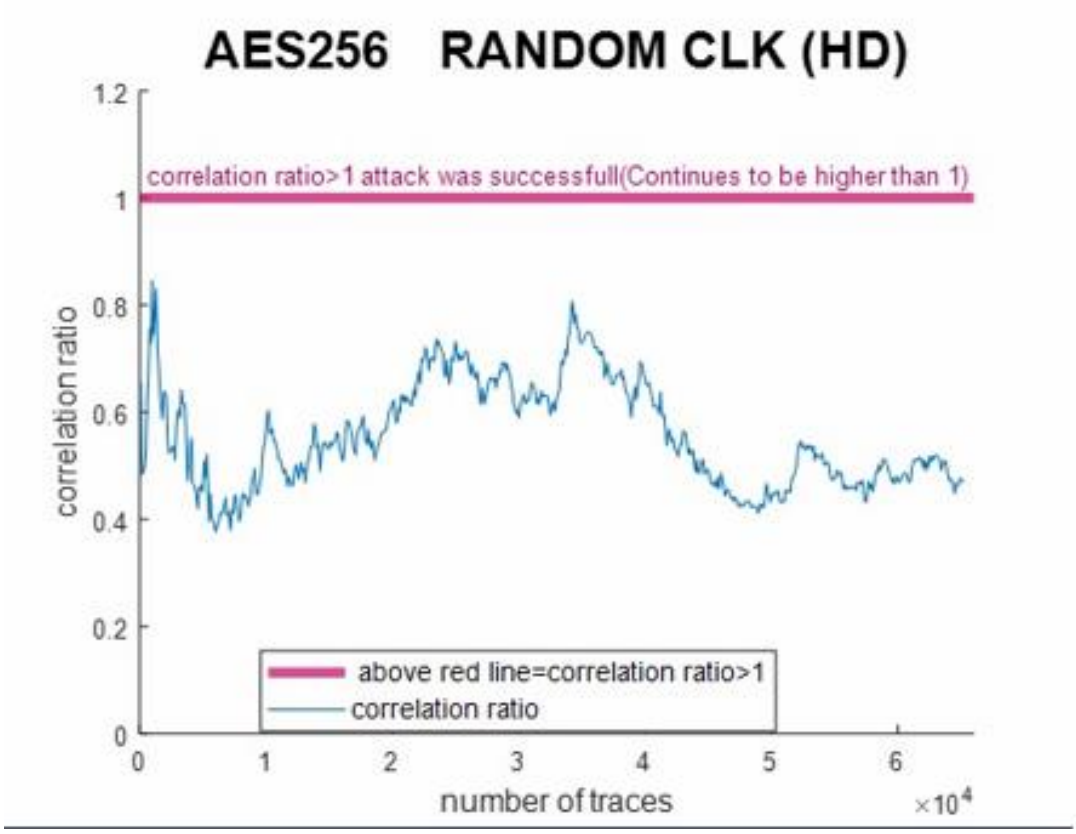
הגרפים 7,8 מתארים את הקורלציה המקסימלית המתקבלת בין ההיפותזות לבין מדידות ההספק על המערכת RandomClockAES המוגנת , עבור 65,280 הצפנות .

מ2 הגרפים ניתן לראות כי ביחס לשאר ההיפותזות , הקורלציה של המפתח האמיתי היא אינה המקסימלית. מכאן ניתן להסיק כי ההתקפה לא הצליחה ומערכת RandomClockAES נשארה מוגנת מהתקפות בשתי השיטות השונות.

עבור המערכת המוגנת בטכנולוגיית הRandomClock, כעת נציג את מדד הCR
(כמתואר בפרק מערך התקיפה) , התלוי בכמות מחזורי ההצפנה הנמדדים .



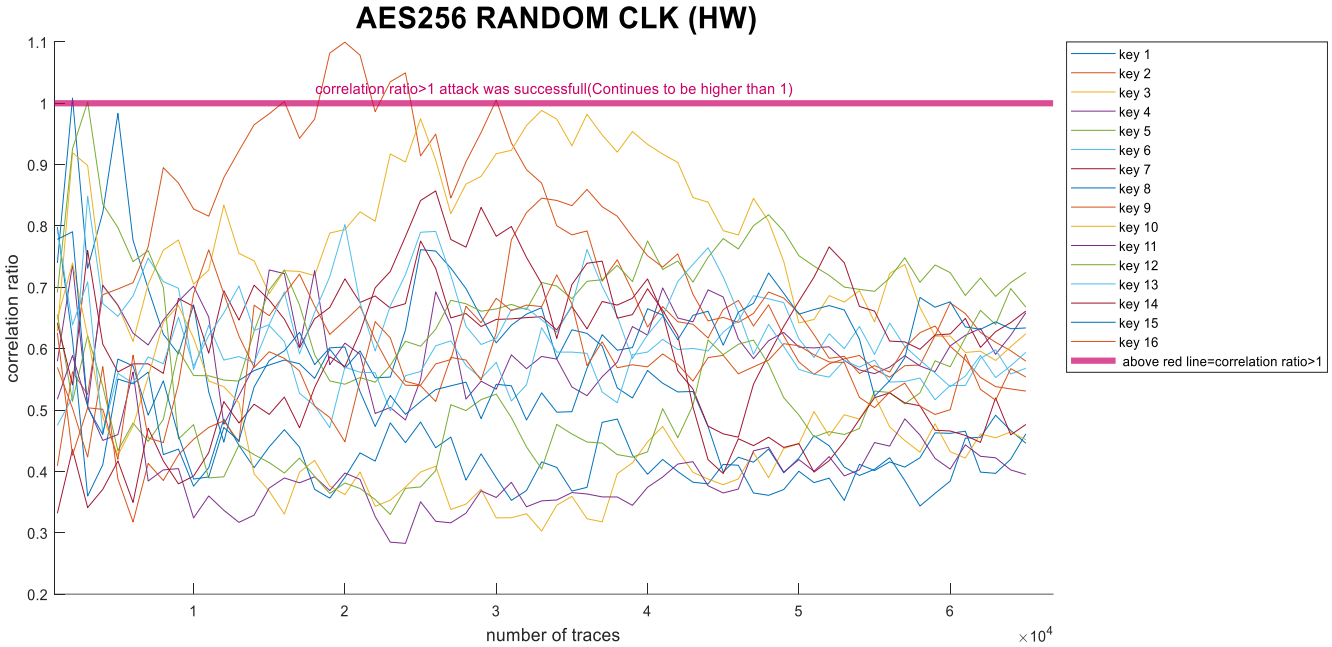
גרף 9 - RandomClock Protected AES256 HW correlation ratio over number of traces



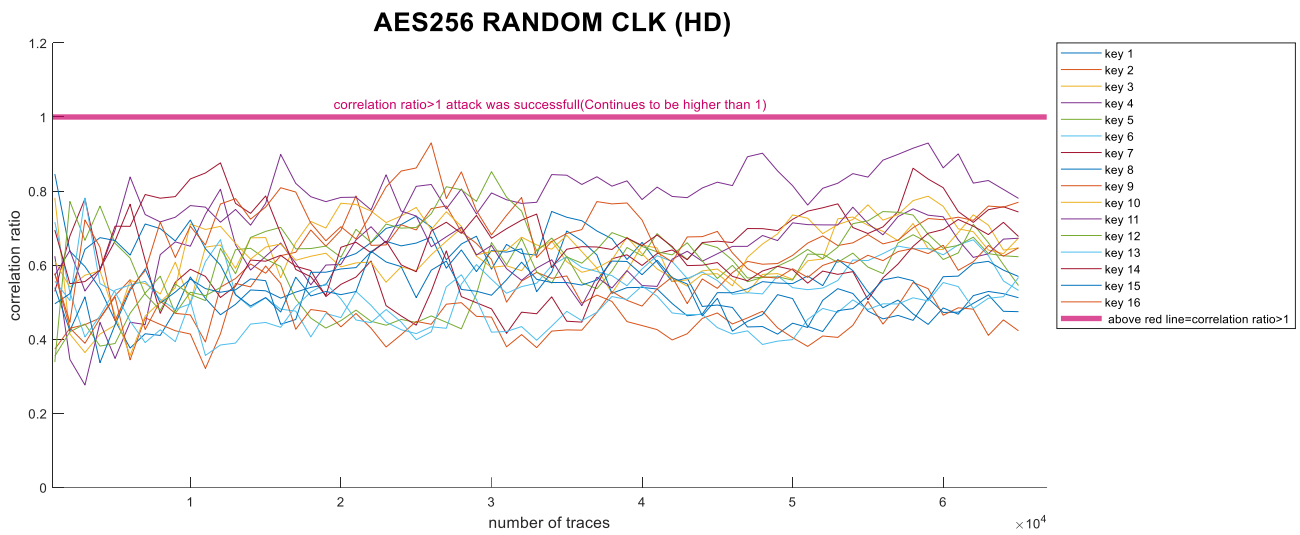
גרף 10 - RandomClock Protected AES256 HD correlation ratio over number of traces

מגרפים 9,10 ניתן לראות כי מדד ה-correlation ratio אינו חוצה את קו ה-1 , לכן אנו נסיק כי המערכת RandomClockAES אכן מוגנת לחלוטין ואינה נפרצת למשך 65,280 הצפנות בשתי שיטות.

לאחר שראינו כי המערכת RandomClockAES אכן נשארת מוגנת תחת השיטות השונות ,
נרצה להציג גרף CR עבור 16 המפתחות .



גרף 11 - RandomClock Protected AES256 HW Correlation ratio for 16 keys over num of traces



גרף 12 - RandomClock Protected AES256 HD Correlation ratio for 16 keys over num of traces

מגרפים 11,12 ניתן לראות כי המערכת RandomClockAES אכן מוגנת עבור 16 המפתחות במשך 65,280 הצפנות . ערכי הקורלציה שעולים ויורדים קצת מעל קו ה-1 אך אינם ממשיכים לעלות וחוזרים חזרה מתחת לקו . זהו מצב שכיח במערכות מוגנות ולכן אנו נוכל להגיד כי מפתחות המערכת RandomClockAES לא נתגלו והמערכת אינה נפרצה .

כעת , נרכז את התוצאות שקיבלנו בטבלה מסכמת :

Type of Implementation	Regular AES	Regular AES	RandomClock AES	RandomClock AES	Improvement	Improvement
Type of attack	HW	HD	HW	HD	HW	HD
Measurement to Disclosure	≈4001	≈301	>65,280	>65,280	>1,531%	>21,584%

טבלה 5 – ריכוז תוצאות המדידות

סיכום ומסקנות :

המטרה העיקרית של הפרויקט שלנו הוא ליצור AES בעל טכנולוגיית RandomClock בכדי לשפר את יעילות האבטחה של ה-AES הרגיל ולהקשות על התוקף בזמן תקיפת ערוץ צד .

בחלק הראשון , בנינו את הפונקציות השונות של ה AES ומימשנו AES "פשוט" ללא תוסף ההגנה כלשהיא , בכדי להבין את רעיון ההצפנה ובכדי להשוות לבסוף את אבטחת המערכת AES "הפשוטה" אל מול מערכת RandomClockAES להעלאת יעילות ההצפנה והאבטחה.

בחלק השני , בנינו את מערכת הRandomClock באמצעות רכיב ה PLL עם הזזת הפאזה ובאמצעות BufferMux גלובלי ו LFSR Select לשינוי שעון העבודה העכשווי של מערך ה AES בצורה PSEODO אקראית . בנוסף יצרנו עוד LFSR אשר "ישלוט" ב- LFSR Select ויקבע לו את זמן העבודה בשעון פאזה 0 (השעון שמפעיל את הLFSR Select הרגיל) כך שהמערכת תוכל לעבוד עם כלל השעונים . כך למעשה ייצרנו 6 שעונים שונים אשר בעזרת הרכיבים השונים , נעבוד למעשה בכל פעם עם שעון אחר . לאחר בניית ה-RandomClock בדקנו שהוא מתפקד היטב בעזרת סימולציה (איור 22).

בחלק הבא , חיברנו בין מערכת ה AES " הפשוט" לבין טכנולוגיית הRandomClock ובכך יצרנו AES משופר (RandomClockAES), אשר בכל פעולה הוא מקבל בכניסתו שעון עבודה אחר ובכך מקשה על פעולת הפריצה , מגביר את יעילות האבטחה , ומקשה על השגת נתונים סטטיסטיים של המערכת בעלת הצופן הסודי .
לפני הצריבה בדקנו שהRandomClockAES עדיין מצפין לאחר סימולציה (איור 25) ראינו כי ההצפנה אכן מתבצעת באופן שוטף ואף בעלת סיבוכיות תקיפה גבוהה יותר .

בחלק האחרון , לאחר בניית מערך ההגנה המשופר , תכננו מערך תקיפה וביצענו תקיפת ערוץ צד על גבי המערכת AES הפשוט ועל RandomClockAES . מהתוצאות ראינו כי הAES הפשוט נפרץ לאחר שיטות מערך התקיפה , ואילו מערכת הRandomClockAES אינה פריצה לכלל ההצפנות (65,280) ולכל שיטות מערך התקיפה .

קיבלנו שיפור משמעותי בתוצאות ההגנה בכ-21,000% בשיטת HD וכ1,500% בשיטת HW (טבלה 5) אך במערכת RandomClockAES זמן כל סבב בהצפנה התארך מ27ns לכדי 133ns וכן התווספו רכיבים כמו PLL , MUX , LFSR .

במערכות כאלו , הדורשות הגנה כגון חברות המחזיקות מידע רגיש, מערכות ביטחוניות , או אפילו בפלאפון האישי שלנו , ערך חשיבות ההצפנה וחוסן ההגנה הוא ערך עליון העומד גם מעל חסרונות כמו עלות גבוה ,צריכת חשמל גבוהה , זמן עבודת ההגנה , תוספות רכיבים וכו'. אי לכך מערכת ההגנה RandomClockAES כמו שלנו אשר משפרת משמעותית את יכולות ההגנה , מציגה פתרון הגנתי משופר מהמערכת RegularAES .

הצעה לעבודות המשך

בפרויקט זה חקרנו קראנו ניסינו ומימשנו את לוגיקת הCLOCK_RANDOM על מערך ההצפנה AES . קיימות דרכים נוספות לשיפור יעילות האבטחה של מערך הצפנת AES שאותן נוכל לשלב ולאחד עם טכנולוגיית הCLOCK _RANDOM שבנינו .

שיטת מיסוך – Masking הינה שיטת התגוננות ידועה להגנה על מימושים של מערכות הצפנת צופן בלוקים מפני התקפות ערוץ צד. העיקרון הוא לפצל באופן אקראי כל משתנה רגיש בחישובי הביניים ל- d + 1 חלקים, כאשר d נקרא סדר ה- Masking.

Frequency Hopping - הכנסת קפיצות תדרים על ידי שינוי אקראי בתדר שעון מערכת ההצפנה, דבר המסייע בשיפור הגנת המערכת מפני התקפות קורלציית הספק.

ניתן להשוות בין השיטות השונות ולבחור את השיטה היעילה ביותר , ואך ניתן לבחון את שיתוף הפעולה של מס שיטות שונות לשיפור משמעותי של מערך ההגנה וההצפנה .

כלים נדרשים :

1. שפות תכנות אפשריות
   1.1. Python
   1.2. MATLAB
   1.3. Verilog
   1.4. C

2. רכיב פיתוח FPGA
   2.1. Xilinx Digilent Zybo Zynq-7000 ARM/FPGA SoC Trainer Board

3. סביבת פיתוח ועיבוד נתונים
   3.1. Microsoft Windows 10
   3.2. Xilinx Vivado® Design Suite
   3.3. Mathworks MATLAB 2019
   3.4. Python

4. ציוד מדידה
   4.1. רכיבים אלקטרונים למימוש החיישן .

- מדידות מסוימות נצטרך לבצע במעבדה באוניברסיטת "בר-אילן" .
- לחלק מהמדידות נוכל להיעזר בציוד הקיים במעבדות של "אפקה" .
- בנוסף קיים סיכוי גדול שמהלך הפרויקט ניאלץ להיעזר באובייקטים נוספים , ולשם כך נצטרך את עזרת מכללת "אפקה".

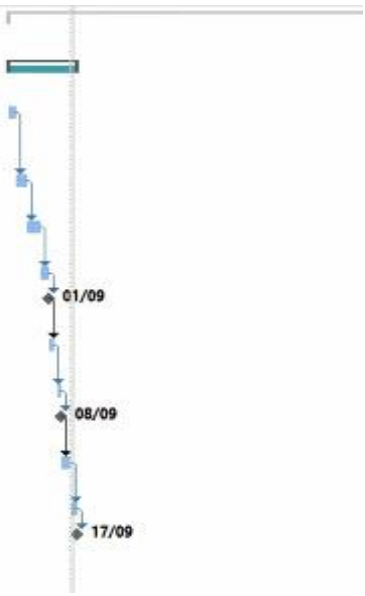
הציוד החסר ניתן להשגה ע"י הצטרפות לתוכנית האקדמית של Xilinx, Xilinx university program, אשר במסגרתו זכאי מוסד אקדמי לקבלת לוח הפיתוח המדובר, כאשר זמן האספקה עומד על מספר חודשים והעלות הינה עבור משלוח ומכס בלבד.

עלות משוערת של רכיב FPGA מסוג זה לפיתוח עצמאי עומד על כבערך $300.

תכנית עבודה:

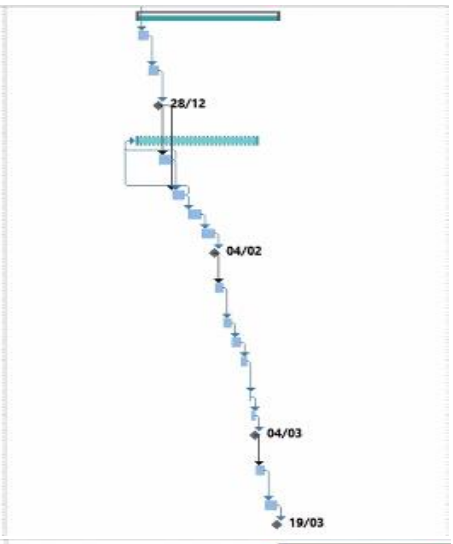
תרשים גאנט מפורט

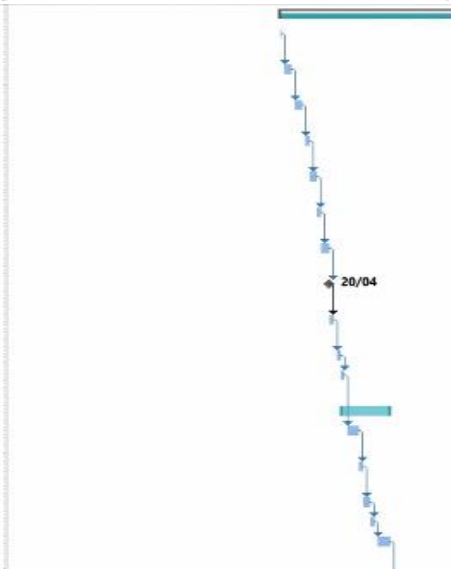
פריקט גמר	242 days	Mon 10/08/20	Tue 13/07/21		
מסמך הגדרת SOW הפרויקט	29 days	Mon 10/08/20	Thu 17/09/20		
מבוא	4 days	Mon 10/08/20	Thu 13/08/20		עוז
מטרות יעדים ומדדים	4 days	Fri 14/08/20	Wed 19/08/20	2	עוז
סקירה ספרותית וסקר שוק	6 days	Thu 20/08/20	Thu 27/08/20	3	עוז ועומר
תרשים בלוקים	3 days	Fri 28/08/20	Tue 01/09/20	4	עומר
פגישה עם המנחה	0 days	Tue 01/09/20	Tue 01/09/20	5	עוז ועומר
אמצעים וכלים נדרשים	3 days	Wed 02/09/20	Fri 04/09/20	6	עומר
תוצר הפרויקט	2 days	Mon 07/09/20	Tue 08/09/20	7	עומר
הגשה לאישור המנחה	0 days	Tue 08/09/20	Tue 08/09/20	8	עוז ועומר
בדיקה של המנחה	4 days	Wed 09/09/20	Mon 14/09/20	9	
תיקונים	3 days	Tue 15/09/20	Thu 17/09/20	10	עוז
הגשה של הגדרת הפרויקט	0 days	Thu 17/09/20	Thu 17/09/20	11	עוז ועומר

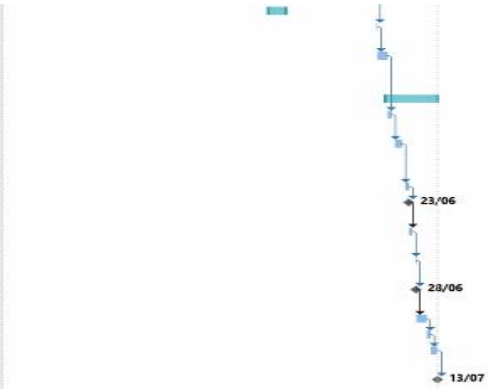


דוח התקדמות(דוח PDRתכנון)	64 days	Thu 17/09/20	Tue 15/12/20		
פגישה עם המנחה	0 days	Thu 17/09/20	Thu 17/09/20		עוז ועומר
למידה עצמית	10 days	Thu 17/09/20	Wed 30/09/20	14	עוז ועומר
בחינת חלופות מערכתיות	5 days	Thu 01/10/20	Wed 07/10/20	15	עוז ועומר
דרכי התמודדות	5 days	Thu 08/10/20	Wed 14/10/20	16	עומר
תיקוני מסמך SOW	5 days	Thu 15/10/20	Wed 21/10/20	17	עוז
תכן ראשוני	7 days	Thu 22/10/20	Fri 30/10/20	18	עומר ועוז
המשך סקירה ספרותית	7 days	Mon 02/11/20	Tue 10/11/20	19	עומר ועוז
רינון שינויים במסמך SOW	2 days	Wed 11/11/20	Thu 12/11/20	20	עומר
פערים	7 days	Fri 13/11/20	Mon 23/11/20	21	עוז
עדכון מקורות	2 days	Tue 24/11/20	Wed 25/11/20	22	עוז
הגשה לאישור המנחה	0 days	Wed 25/11/20	Wed 25/11/20	23	עומר
בדיקה של המנחה	8 days	Thu 26/11/20	Mon 07/12/20	24	
תיקונים	4 days	Tue 08/12/20	Fri 11/12/20	25	עוז ועומר
הגשת דוח התקדמות PDR	1 day	Mon 14/12/20	Mon 14/12/20	26	עוז ועומר

	דוח הביניים	70 days	Tue 15/12/20	Sat 20/03/21		
	תקציר מנהלים	5 days	Tue 15/12/20	Mon 21/12/20	27	עומר
	סקירה ספרותית וסקר שוק	5 days	Tue 22/12/20	Mon 28/12/20	29	עוז
	פגישת מנחה	0 days	Mon 28/12/20	Mon 28/12/20	30	עומר ועוז
	תכן מפורט	59 days	Tue 15/12/20	Fri 05/03/21	33,34	עוז
	פירוט והרחבה של מרכיבי הפרויקט	7 days	Tue 29/12/20	Wed 06/01/21	31	עומר
	גרפים וחישובים	7 days	Thu 07/01/21	Fri 15/01/21	31,33	עומר
	הצגת סימולציות	7 days	Mon 18/01/21	Tue 26/01/21	34	עוז
	פירוט האלגוריתם	7 days	Wed 27/01/21	Thu 04/02/21	35	עוז
	פגישה עם המנחה	0 days	Thu 04/02/21	Thu 04/02/21	36	עומר ועוז
	עדכון תכנית העבודה	4 days	Fri 05/02/21	Wed 10/02/21	37	עומר ועוז
	ריכוז שינויים	4 days	Thu 11/02/21	Tue 16/02/21	38	עומר
	סיכונים	4 days	Wed 17/02/21	Mon 22/02/21	39	עומר
	עדכון דרכי התמודדות	4 days	Tue 23/02/21	Fri 26/02/21	40	עוז ועומר
	עדכון מקורות	1 day	Mon 01/03/21	Mon 01/03/21	41	עומר ועוז
	נספחים	3 days	Tue 02/03/21	Thu 04/03/21	42	עומר ועוז
	פגישה עם המנחה	0 days	Thu 04/03/21	Thu 04/03/21	43	עומר ועוז
	בדיקה של המנחה	4 days	Fri 05/03/21	Wed 10/03/21	44	
	תיקונים	7 days	Thu 11/03/21	Fri 19/03/21	45	עומר ועוז
	הגשת דוח ביניים	0 days	Fri 19/03/21	Fri 19/03/21	46	עוז
	דוח סופי	83 days	Fri 19/03/21	Tue 13/07/21		
	תודות	1 day	Fri 19/03/21	Fri 19/03/21		עוז ועומר
	איורים טבלאות מסחאות וגרפים	5 days	Mon 22/03/21	Fri 26/03/21	49	עוז ועומר
	תקציר מנהלים	5 days	Mon 29/03/21	Fri 02/04/21	50	עוז
	תקציר מנהלים באנגלית	3 days	Mon 05/04/21	Wed 07/04/21	51	עומר
	מילון מונחים	3 days	Thu 08/04/21	Mon 12/04/21	52	עומר ועוז
	מבוא	3 days	Tue 13/04/21	Thu 15/04/21	53	עומר
	מטרות ויעדים	3 days	Fri 16/04/21	Tue 20/04/21	54	עומר
	פגישה עם המנחה	0 days	Tue 20/04/21	Tue 20/04/21	55	עומר ועוז
	סקירה ספרותית	3 days	Wed 21/04/21	Fri 23/04/21	56	עוז
	סקר שוק	3 days	Mon 26/04/21	Wed 28/04/21	57	עומר
	עדכון חלופות מהדוח ההנדסי	2 days	Thu 29/04/21	Fri 30/04/21	58	עומר ועוז
	תכן מפורט	22 days	Fri 30/04/21	Mon 31/05/21		
	פירוט והרחבה על מרכיבי הפרויקט	6 days	Mon 03/05/21	Mon 10/05/21	59	עומר ועוז
	תרשים מלבנים משודכן	3 days	Tue 11/05/21	Thu 13/05/21	61	עוז
	חישובים	3 days	Fri 14/05/21	Tue 18/05/21	62	עוז
	סימולציות נוספות	3 days	Wed 19/05/21	Fri 21/05/21	63	עומר
	הסבר על האלגוריתם	6 days	Mon 24/05/21	Mon 31/05/21	64	עומר ועוז
	תוצר הפרויקט	8 days	Fri 19/03/21	Tue 30/03/21		
	הצגת התוצר למנחה	1 day	Tue 01/06/21	Tue 01/06/21	65	עומר ועוז
	נספחים : קודי האלגוריתם והתוכנה	5 days	Wed 02/06/21	Tue 08/06/21	67	עומר ועוז
	בדיקות והערכה	26 days	Tue 08/06/21	Tue 13/07/21		
	בדיקת מודלים ואישורם	3 days	Wed 09/06/21	Fri 11/06/21	68	עוז
	בדיקת האלגוריתם ויישומו	5 days	Mon 14/06/21	Fri 18/06/21	70	עומר
	סיכום ומסקנות	3 days	Mon 21/06/21	Wed 23/06/21	71	עומר ועוז
	פגישה עם המנחה	0 days	Wed 23/06/21	Wed 23/06/21	72	עומר ועוז
	עדכון ניהול סיכונים	2 days	Thu 24/06/21	Fri 25/06/21	73	עומר
	הצעה לעבודת המשך	1 day	Mon 28/06/21	Mon 28/06/21	74	עוז
	הגשת לאישור המנחה	0 days	Mon 28/06/21	Mon 28/06/21	75	עומר
	בדיקת המנחה	5 days	Tue 29/06/21	Mon 05/07/21	76	
	תיקונים	3 days	Tue 06/07/21	Thu 08/07/21	77	עומר ועוז
	הדפסת ספר הפרויקט	3 days	Fri 09/07/21	Tue 13/07/21	78	עומר ועוז
	הגשת הפרויקט	0 days	Tue 13/07/21	Tue 13/07/21	79	עומר ועוז







ניהול סיכונים :

1) נגיף הקורונה עדיין נמצא ומביא איתו סיכונים כמו בידודים וחוסר אינטרקציה בין השותפים למכללה , דבר היכול למנוע ביצוע מדידות במעבדה / עיכוב בלקיחת רכיבים מהמכללה .
2) קיימת האפשרות שסביבת העבודה החינמית לא תספיק למהלך העבודה , ונצטרך לעבור לסביבה המתקדמת בה אנו נדרשים להיות עם רישיון.
3) סיכון נוסף , יתכנו ויתגלו סיכונים נוספים בהמשך .

למרות הקורונה וסיכונים נוספים , הצלחנו להתמודד מול סיכונים אלו , לאסוף את דגימות המדידות ולסיים את תהליך הפרויקט.

רשימת מקורות :

קריפטוגרפיה – ויקיפדיה :

https://he.wikipedia.org/wiki/%D7%A7%D7%A8%D7%99%D7%A4%D7%98%D7%95%D7%92%D7%A8%D7%A4%D7%99%D7%94#%D7%A8%D7%A7%D7%A2_%D7%94%D7%99%D7%A1%D7%98%D7%95%D7%A8%D7%99

AES – ויקיפדיה :

https://he.wikipedia.org/wiki/Advanced_Encryption_Standard

ספר נוסף על AES:

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Rambus :

https://www.rambus.com/security/dpa-countermeasures/dpa-resistant-core/anti-/tamper-cryptographic-cores

ספר על התקפות צד – Christof Paar :

https://www.emsec.ruhr-uni-bochum.de/media/attachments/files/2015/09/IKV-1_2015-04-28.pdf

קישור להרצאות של Christof Paar :

https://www.youtube.com/channel/UC1usFRN4LCMcfIV7UjHNuQg/videos

קריפטוגרפיה וקריפטואנליזה :

http://swarm.cs.pub.ro/~mbarbulescu/cripto/Understanding%20Cryptography%20by%20Christof%20Paar%20.pdf

דפי יצרן של כרטיס ה FPGA :

https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/reference-manual?_ga=2.29122165.811589740.1565549428-168987004.1565549428

מעבדות לתרגול עם סביבת הפיתוח של Vivado :

https://www.xilinx.com/support/university/vivado/vivado-teaching-material/hdl-design.html

מחשבון אלגוריתם הצפנת ה-AES :

AES Encryption – Easily encrypt or decrypt strings or files (online-domain-tools.com)

A Novel Countermeasure to Resist Side Channel Attacks on FPGA Implementations

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.686.2677&rep=rep1&type=pdf

Random clocking induced DPA attack immunity in FPGAs

https://ieeexplore.ieee.org/document/5472570

A PLL-Based Spread-Spectrum Clock Generator With a Ditherless Fractional Topolog

http://140.120.31.200/Pic/Writings/1917_200901TCASI_sscg.pdf

A Highly Efficient Power Model for Correlation Power Analysis (CPA) of Pipelined Advanced Encryption Standard (AES)

https://ieeexplore.ieee.org/document/9180778

נספחים :

Python outcome :

AES Encipher tests

```
0x2b7e1516, 0x28aed2a6, 0xabf71588, 0x09cf4f3c
0x6bc1bee2, 0x2e409f96, 0xe93d7e11, 0x7393172a
Doing the 128 bit key expansion
Doing the 128 bit key expansion []=round_keys
Inside substw:
b0 = 0xcf, b1 = 0x4f, b2 = 0x3c, b3 = 0x09
s0 = 0x8a, s1 = 0x84, s2 = 0xeb, s3 = 0x01
res = 0x8a84eb01
Inside next 128bit key:
w0 = 0x2b7e1516, w1 = 0x28aed2a6, w2 = 0xabf71588, w3 = 0x09cf4f3c
rol = 0xcf4f3c09, subst = 0x8a84eb01, rcon = 0x01, t = 0x8b84eb01
k0 = 0xa0fafe17, k1 = 0x88542cb1, k2 = 0x23a33939, k3 = 0x2a6c7605
Inside substw:
b0 = 0x6c, b1 = 0x76, b2 = 0x05, b3 = 0x2a
s0 = 0x50, s1 = 0x38, s2 = 0x6b, s3 = 0xe5
res = 0x50386be5
Inside next 128bit key:
w0 = 0xa0fafe17, w1 = 0x88542cb1, w2 = 0x23a33939, w3 = 0x2a6c7605
rol = 0x6c76052a, subst = 0x50386be5, rcon = 0x02, t = 0x52386be5
k0 = 0xf2c295f2, k1 = 0x7a96b943, k2 = 0x5935807a, k3 = 0x7359f67f
Inside substw:
b0 = 0x59, b1 = 0xf6, b2 = 0x7f, b3 = 0x73
s0 = 0xcb, s1 = 0x42, s2 = 0xd2, s3 = 0x8f
res = 0xcb42d28f
Inside next 128bit key:
w0 = 0xf2c295f2, w1 = 0x7a96b943, w2 = 0x5935807a, w3 = 0x7359f67f
rol = 0x59f67f73, subst = 0xcb42d28f, rcon = 0x04, t = 0xcf42d28f
k0 = 0x3d80477d, k1 = 0x4716fe3e, k2 = 0x1e237e44, k3 = 0x6d7a883b
Inside substw:
b0 = 0x7a, b1 = 0x88, b2 = 0x3b, b3 = 0x6d
s0 = 0xda, s1 = 0xc4, s2 = 0xe2, s3 = 0x3c
res = 0xdac4e23c
Inside next 128bit key:
w0 = 0x3d80477d, w1 = 0x4716fe3e, w2 = 0x1e237e44, w3 = 0x6d7a883b
rol = 0x7a883b6d, subst = 0xdac4e23c, rcon = 0x08, t = 0xd2c4e23c
k0 = 0xef44a541, k1 = 0xa8525b7f, k2 = 0xb671253b, k3 = 0xdb0bad00
Inside substw:
b0 = 0x0b, b1 = 0xad, b2 = 0x00, b3 = 0xdb
s0 = 0x2b, s1 = 0x95, s2 = 0x63, s3 = 0xb9
res = 0x2b9563b9
Inside next 128bit key:
w0 = 0xef44a541, w1 = 0xa8525b7f, w2 = 0xb671253b, w3 = 0xdb0bad00
rol = 0x0bad00db, subst = 0x2b9563b9, rcon = 0x10, t = 0x3b9563b9
k0 = 0xd4d1c6f8, k1 = 0x7c839d87, k2 = 0xcaf2b8bc, k3 = 0x11f915bc
Inside substw:
b0 = 0xf9, b1 = 0x15, b2 = 0xbc, b3 = 0x11
s0 = 0x99, s1 = 0x59, s2 = 0x65, s3 = 0x82
res = 0x99596582
Inside next 128bit key:
w0 = 0xd4d1c6f8, w1 = 0x7c839d87, w2 = 0xcaf2b8bc, w3 = 0x11f915bc
rol = 0xf915bc11, subst = 0x99596582, rcon = 0x20, t = 0xb9596582
k0 = 0x6d88a37a, k1 = 0x110b3efd, k2 = 0xdbf98641, k3 = 0xca0093fd
Inside substw:
b0 = 0x00, b1 = 0x93, b2 = 0xfd, b3 = 0xca
s0 = 0x63, s1 = 0xdc, s2 = 0x54, s3 = 0x74
res = 0x63dc5474
Inside next 128bit key:
w0 = 0x6d88a37a, w1 = 0x110b3efd, w2 = 0xdbf98641, w3 = 0xca0093fd
rol = 0x0093fdca, subst = 0x63dc5474, rcon = 0x40, t = 0x23dc5474
```

k0 = 0x4e54f70e, k1 = 0x5f5fc9f3, k2 = 0x84a64fb2, k3 = 0x4ea6dc4f
Inside substw:
b0 = 0xa6, b1 = 0xdc, b2 = 0x4f, b3 = 0x4e
s0 = 0x24, s1 = 0x86, s2 = 0x84, s3 = 0x2f
res = 0x2486842f
Inside next 128bit key:
w0 = 0x4e54f70e, w1 = 0x5f5fc9f3, w2 = 0x84a64fb2, w3 = 0x4ea6dc4f
rol = 0xa6dc4f4e, subst = 0x2486842f, rcon = 0x80, t = 0xa486842f
k0 = 0xead27321, k1 = 0xb58dbad2, k2 = 0x312bf560, k3 = 0x7f8d292f
Inside substw:
b0 = 0x8d, b1 = 0x29, b2 = 0x2f, b3 = 0x7f
s0 = 0x5d, s1 = 0xa5, s2 = 0x15, s3 = 0xd2
res = 0x5da515d2
Inside next 128bit key:
w0 = 0xead27321, w1 = 0xb58dbad2, w2 = 0x312bf560, w3 = 0x7f8d292f
rol = 0x8d292f7f, subst = 0x5da515d2, rcon = 0x1b, t = 0x46a515d2
k0 = 0xac7766f3, k1 = 0x19fadc21, k2 = 0x28d12941, k3 = 0x575c006e
Inside substw:
b0 = 0x5c, b1 = 0x00, b2 = 0x6e, b3 = 0x57
s0 = 0x4a, s1 = 0x63, s2 = 0x9f, s3 = 0x5b
res = 0x4a639f5b
Inside next 128bit key:
w0 = 0xac7766f3, w1 = 0x19fadc21, w2 = 0x28d12941, w3 = 0x575c006e
rol = 0x5c006e57, subst = 0x4a639f5b, rcon = 0x36, t = 0x7c639f5b
k0 = 0xd014f9a8, k1 = 0xc9ee2589, k2 = 0xe13f0cc8, k3 = 0xb6630ca6
Input key:
0x2b7e1516, 0x28aed2a6, 0xabf71588, 0x09cf4f3c

Generated keys:
0x2b7e1516, 0x28aed2a6, 0xabf71588, 0x09cf4f3c
0xa0fafe17, 0x88542cb1, 0x23a33939, 0x2a6c7605
0xf2c295f2, 0x7a96b943, 0x5935807a, 0x7359f67f
0x3d80477d, 0x4716fe3e, 0x1e237e44, 0x6d7a883b
0xef44a541, 0xa8525b7f, 0xb671253b, 0xdb0bad00
0xd4d1c6f8, 0x7c839d87, 0xcaf2b8bc, 0x11f915bc
0x6d88a37a, 0x110b3efd, 0xdbf98641, 0xca0093fd
0x4e54f70e, 0x5f5fc9f3, 0x84a64fb2, 0x4ea6dc4f
0xead27321, 0xb58dbad2, 0x312bf560, 0x7f8d292f
0xac7766f3, 0x19fadc21, 0x28d12941, 0x575c006e
0xd014f9a8, 0xc9ee2589, 0xe13f0cc8, 0xb6630ca6

Initial AddRoundKeys round.
AddRoundKey key, block in and block out:
0x2b7e1516, 0x28aed2a6, 0xabf71588, 0x09cf4f3c
0x6bc1bee2, 0x2e409f96, 0xe93d7e11, 0x7393172a
0x40bfabf4, 0x06ee4d30, 0x42ca6b99, 0x7a5c5816

Round 01

Inside substw:
b0 = 0x40, b1 = 0xbf, b2 = 0xab, b3 = 0xf4
s0 = 0x09, s1 = 0x08, s2 = 0x62, s3 = 0xbf
res = 0x090862bf
Inside substw:
b0 = 0x06, b1 = 0xee, b2 = 0x4d, b3 = 0x30
s0 = 0x6f, s1 = 0x28, s2 = 0xe3, s3 = 0x04
res = 0x6f28e304
Inside substw:
b0 = 0x42, b1 = 0xca, b2 = 0x6b, b3 = 0x99
s0 = 0x2c, s1 = 0x74, s2 = 0x7f, s3 = 0xee
res = 0x2c747fee
Inside substw:
b0 = 0x7a, b1 = 0x5c, b2 = 0x58, b3 = 0x16
s0 = 0xda, s1 = 0x4a, s2 = 0x6a, s3 = 0x47
res = 0xda4a6a47

```
SubBytes block in and block out:
0x40bfabf4, 0x06ee4d30, 0x42ca6b99, 0x7a5c5816
0x090862bf, 0x6f28e304, 0x2c747fee, 0xda4a6a47

ShiftRows block in and block out:
0x090862bf, 0x6f28e304, 0x2c747fee, 0xda4a6a47
0x09287f47, 0x6f746abf, 0x2c4a6204, 0xda08e3ee

MixColumns block in and block out:
0x09287f47, 0x6f746abf, 0x2c4a6204, 0xda08e3ee
0x529f16c2, 0x978615ca, 0xe01aae54, 0xba1a2659

AddRoundKey key, block in and block out:
0xa0fafe17, 0x88542cb1, 0x23a33939, 0x2a6c7605
0x529f16c2, 0x978615ca, 0xe01aae54, 0xba1a2659
0xf265e8d5, 0x1fd2397b, 0xc3b9976d, 0x9076505c


 Round 02
 ---------
Inside substw:
b0 = 0xf2, b1 = 0x65, b2 = 0xe8, b3 = 0xd5
s0 = 0x89, s1 = 0x4d, s2 = 0x9b, s3 = 0x03
res = 0x894d9b03
Inside substw:
b0 = 0x1f, b1 = 0xd2, b2 = 0x39, b3 = 0x7b
s0 = 0xc0, s1 = 0xb5, s2 = 0x12, s3 = 0x21
res = 0xc0b51221
Inside substw:
b0 = 0xc3, b1 = 0xb9, b2 = 0x97, b3 = 0x6d
s0 = 0x2e, s1 = 0x56, s2 = 0x88, s3 = 0x3c
res = 0x2e56883c
Inside substw:
b0 = 0x90, b1 = 0x76, b2 = 0x50, b3 = 0x5c
s0 = 0x60, s1 = 0x38, s2 = 0x53, s3 = 0x4a
res = 0x6038534a
SubBytes block in and block out:
0xf265e8d5, 0x1fd2397b, 0xc3b9976d, 0x9076505c
0x894d9b03, 0xc0b51221, 0x2e56883c, 0x6038534a

ShiftRows block in and block out:
0x894d9b03, 0xc0b51221, 0x2e56883c, 0x6038534a
0x89b5884a, 0xc0565303, 0x2e389b21, 0x604d123c

MixColumns block in and block out:
0x89b5884a, 0xc0565303, 0x2e389b21, 0x604d123c
0x0f31e929, 0x319a3558, 0xaec95893, 0x39f04d87

AddRoundKey key, block in and block out:
0xf2c295f2, 0x7a96b943, 0x5935807a, 0x7359f67f
0x0f31e929, 0x319a3558, 0xaec95893, 0x39f04d87
0xfdf37cdb, 0x4b0c8c1b, 0xf7fcd8e9, 0x4aa9bbf8


 Round 03
 ---------
Inside substw:
b0 = 0xfd, b1 = 0xf3, b2 = 0x7c, b3 = 0xdb
s0 = 0x54, s1 = 0x0d, s2 = 0x10, s3 = 0xb9
res = 0x540d10b9
Inside substw:
b0 = 0x4b, b1 = 0x0c, b2 = 0x8c, b3 = 0x1b
s0 = 0xb3, s1 = 0xfe, s2 = 0x64, s3 = 0xaf
res = 0xb3fe64af
Inside substw:
b0 = 0xf7, b1 = 0xfc, b2 = 0xd8, b3 = 0xe9
```

s0 = 0x68, s1 = 0xb0, s2 = 0x61, s3 = 0x1e
res = 0x68b0611e
Inside substw:
b0 = 0x4a, b1 = 0xa9, b2 = 0xbb, b3 = 0xf8
s0 = 0xd6, s1 = 0xd3, s2 = 0xea, s3 = 0x41
res = 0xd6d3ea41
SubBytes block in and block out:
0xfdf37cdb, 0x4b0c8c1b, 0xf7fcd8e9, 0x4aa9bbf8
0x540d10b9, 0xb3fe64af, 0x68b0611e, 0xd6d3ea41

ShiftRows block in and block out:
0x540d10b9, 0xb3fe64af, 0x68b0611e, 0xd6d3ea41
0x54fe6141, 0xb3b0eab9, 0x68d310af, 0xd60d641e

MixColumns block in and block out:
0x54fe6141, 0xb3b0eab9, 0x68d310af, 0xd60d641e
0x9151abe1, 0xe5541cfd, 0x014a713e, 0xda7e3134

AddRoundKey key, block in and block out:
0x3d80477d, 0x4716fe3e, 0x1e237e44, 0x6d7a883b
0x9151abe1, 0xe5541cfd, 0x014a713e, 0xda7e3134
0xacd1ec9c, 0xa242e2c3, 0x1f690f7a, 0xb704b90f

Round 04

Inside substw:
b0 = 0xac, b1 = 0xd1, b2 = 0xec, b3 = 0x9c
s0 = 0x91, s1 = 0x3e, s2 = 0xce, s3 = 0xde
res = 0x913ecede
Inside substw:
b0 = 0xa2, b1 = 0x42, b2 = 0xe2, b3 = 0xc3
s0 = 0x3a, s1 = 0x2c, s2 = 0x98, s3 = 0x2e
res = 0x3a2c982e
Inside substw:
b0 = 0x1f, b1 = 0x69, b2 = 0x0f, b3 = 0x7a
s0 = 0xc0, s1 = 0xf9, s2 = 0x76, s3 = 0xda
res = 0xc0f976da
Inside substw:
b0 = 0xb7, b1 = 0x04, b2 = 0xb9, b3 = 0x0f
s0 = 0xa9, s1 = 0xf2, s2 = 0x56, s3 = 0x76
res = 0xa9f25676
SubBytes block in and block out:
0xacd1ec9c, 0xa242e2c3, 0x1f690f7a, 0xb704b90f
0x913ecede, 0x3a2c982e, 0xc0f976da, 0xa9f25676

ShiftRows block in and block out:
0x913ecede, 0x3a2c982e, 0xc0f976da, 0xa9f25676
0x912c7676, 0x3af956de, 0xc0f2ce2e, 0xa93e98da

MixColumns block in and block out:
0x912c7676, 0x3af956de, 0xc0f2ce2e, 0xa93e98da
0x4d25cb1e, 0xecf71646, 0x7658c73b, 0x49bcc9e9

AddRoundKey key, block in and block out:
0xef44a541, 0xa8525b7f, 0xb671253b, 0xdb0bad00
0x4d25cb1e, 0xecf71646, 0x7658c73b, 0x49bcc9e9
0xa2616e5f, 0x44a54d39, 0xc029e200, 0x92b764e9

Round 05

Inside substw:
b0 = 0xa2, b1 = 0x61, b2 = 0x6e, b3 = 0x5f
s0 = 0x3a, s1 = 0xef, s2 = 0x9f, s3 = 0xcf
res = 0x3aef9fcf

```
Inside substw:
b0 = 0x44, b1 = 0xa5, b2 = 0x4d, b3 = 0x39
s0 = 0x1b, s1 = 0x06, s2 = 0xe3, s3 = 0x12
res = 0x1b06e312
Inside substw:
b0 = 0xc0, b1 = 0x29, b2 = 0xe2, b3 = 0x00
s0 = 0xba, s1 = 0xa5, s2 = 0x98, s3 = 0x63
res = 0xbaa59863
Inside substw:
b0 = 0x92, b1 = 0xb7, b2 = 0x64, b3 = 0xe9
s0 = 0x4f, s1 = 0xa9, s2 = 0x43, s3 = 0x1e
res = 0x4fa9431e
SubBytes block in and block out:
0xa2616e5f, 0x44a54d39, 0xc029e200, 0x92b764e9
0x3aef9fcf, 0x1b06e312, 0xbaa59863, 0x4fa9431e

ShiftRows block in and block out:
0x3aef9fcf, 0x1b06e312, 0xbaa59863, 0x4fa9431e
0x3a06981e, 0x1ba543cf, 0xbaa99f12, 0x4fefe363

MixColumns block in and block out:
0x3a06981e, 0x1ba543cf, 0xbaa99f12, 0x4fefe363
0xf89b35ec, 0x4e40724e, 0x025b00c7, 0x34d7d81b

AddRoundKey key, block in and block out:
0xd4d1c6f8, 0x7c839d87, 0xcaf2b8bc, 0x11f915bc
0xf89b35ec, 0x4e40724e, 0x025b00c7, 0x34d7d81b
0x2c4af314, 0x32c3efc9, 0xc8a9b87b, 0x252ecda7


 Round 06
 ---------
Inside substw:
b0 = 0x2c, b1 = 0x4a, b2 = 0xf3, b3 = 0x14
s0 = 0x71, s1 = 0xd6, s2 = 0x0d, s3 = 0xfa
res = 0x71d60dfa
Inside substw:
b0 = 0x32, b1 = 0xc3, b2 = 0xef, b3 = 0xc9
s0 = 0x23, s1 = 0x2e, s2 = 0xdf, s3 = 0xdd
res = 0x232edfdd
Inside substw:
b0 = 0xc8, b1 = 0xa9, b2 = 0xb8, b3 = 0x7b
s0 = 0xe8, s1 = 0xd3, s2 = 0x6c, s3 = 0x21
res = 0xe8d36c21
Inside substw:
b0 = 0x25, b1 = 0x2e, b2 = 0xcd, b3 = 0xa7
s0 = 0x3f, s1 = 0x31, s2 = 0xbd, s3 = 0x5c
res = 0x3f31bd5c
SubBytes block in and block out:
0x2c4af314, 0x32c3efc9, 0xc8a9b87b, 0x252ecda7
0x71d60dfa, 0x232edfdd, 0xe8d36c21, 0x3f31bd5c

ShiftRows block in and block out:
0x71d60dfa, 0x232edfdd, 0xe8d36c21, 0x3f31bd5c
0x712e6c5c, 0x23d3bdfa, 0xe8310ddd, 0x3fd6df21

MixColumns block in and block out:
0x712e6c5c, 0x23d3bdfa, 0xe8310ddd, 0x3fd6df21
0xa0c56369, 0x6fb884e4, 0x4840bfbe, 0xe1d32f0a

AddRoundKey key, block in and block out:
0x6d88a37a, 0x110b3efd, 0xdbf98641, 0xca0093fd
0xa0c56369, 0x6fb884e4, 0x4840bfbe, 0xe1d32f0a
0xcd4dc013, 0x7eb3ba19, 0x93b939ff, 0x2bd3bcf7
```

```
 Round 07
 ---------
Inside substw:
b0 = 0xcd, b1 = 0x4d, b2 = 0xc0, b3 = 0x13
s0 = 0xbd, s1 = 0xe3, s2 = 0xba, s3 = 0x7d
res = 0xbde3ba7d
Inside substw:
b0 = 0x7e, b1 = 0xb3, b2 = 0xba, b3 = 0x19
s0 = 0xf3, s1 = 0x6d, s2 = 0xf4, s3 = 0xd4
res = 0xf36df4d4
Inside substw:
b0 = 0x93, b1 = 0xb9, b2 = 0x39, b3 = 0xff
s0 = 0xdc, s1 = 0x56, s2 = 0x12, s3 = 0x16
res = 0xdc561216
Inside substw:
b0 = 0x2b, b1 = 0xd3, b2 = 0xbc, b3 = 0xf7
s0 = 0xf1, s1 = 0x66, s2 = 0x65, s3 = 0x68
res = 0xf1666568
SubBytes block in and block out:
0xcd4dc013, 0x7eb3ba19, 0x93b939ff, 0x2bd3bcf7
0xbde3ba7d, 0xf36df4d4, 0xdc561216, 0xf1666568

ShiftRows block in and block out:
0xbde3ba7d, 0xf36df4d4, 0xdc561216, 0xf1666568
0xbd6d1268, 0xf356657d, 0xdc66bad4, 0xf1e3f416

MixColumns block in and block out:
0xbd6d1268, 0xf356657d, 0xdc66bad4, 0xf1e3f416
0xac394c73, 0x1f8de8c7, 0x6711b210, 0x253ddb33

AddRoundKey key, block in and block out:
0x4e54f70e, 0x5f5fc9f3, 0x84a64fb2, 0x4ea6dc4f
0xac394c73, 0x1f8de8c7, 0x6711b210, 0x253ddb33
0xe26dbb7d, 0x40d22134, 0xe3b7fda2, 0x6b9b077c


 Round 08
 ---------
Inside substw:
b0 = 0xe2, b1 = 0x6d, b2 = 0xbb, b3 = 0x7d
s0 = 0x98, s1 = 0x3c, s2 = 0xea, s3 = 0xff
res = 0x983ceaff
Inside substw:
b0 = 0x40, b1 = 0xd2, b2 = 0x21, b3 = 0x34
s0 = 0x09, s1 = 0xb5, s2 = 0xfd, s3 = 0x18
res = 0x09b5fd18
Inside substw:
b0 = 0xe3, b1 = 0xb7, b2 = 0xfd, b3 = 0xa2
s0 = 0x11, s1 = 0xa9, s2 = 0x54, s3 = 0x3a
res = 0x11a9543a
Inside substw:
b0 = 0x6b, b1 = 0x9b, b2 = 0x07, b3 = 0x7c
s0 = 0x7f, s1 = 0x14, s2 = 0xc5, s3 = 0x10
res = 0x7f14c510
SubBytes block in and block out:
0xe26dbb7d, 0x40d22134, 0xe3b7fda2, 0x6b9b077c
0x983ceaff, 0x09b5fd18, 0x11a9543a, 0x7f14c510

ShiftRows block in and block out:
0x983ceaff, 0x09b5fd18, 0x11a9543a, 0x7f14c510
0x98b55410, 0x09a9c5ff, 0x1114ea18, 0x7f3cfd3a

MixColumns block in and block out:
0x98b55410, 0x09a9c5ff, 0x1114ea18, 0x7f3cfd3a
0xab05b572, 0xc8eb2b92, 0xec04e2fd, 0x7d21ec34
```

```
AddRoundKey key, block in and block out:
0xead27321, 0xb58dbad2, 0x312bf560, 0x7f8d292f
0xab05b572, 0xc8eb2b92, 0xec04e2fd, 0x7d21ec34
0x41d7c653, 0x7d669140, 0xdd2f179d, 0x02acc51b


 Round 09
 ---------
Inside substw:
b0 = 0x41, b1 = 0xd7, b2 = 0xc6, b3 = 0x53
s0 = 0x83, s1 = 0x0e, s2 = 0xb4, s3 = 0xed
res = 0x830eb4ed
Inside substw:
b0 = 0x7d, b1 = 0x66, b2 = 0x91, b3 = 0x40
s0 = 0xff, s1 = 0x33, s2 = 0x81, s3 = 0x09
res = 0xff338109
Inside substw:
b0 = 0xdd, b1 = 0x2f, b2 = 0x17, b3 = 0x9d
s0 = 0xc1, s1 = 0x15, s2 = 0xf0, s3 = 0x5e
res = 0xc115f05e
Inside substw:
b0 = 0x02, b1 = 0xac, b2 = 0xc5, b3 = 0x1b
s0 = 0x77, s1 = 0x91, s2 = 0xa6, s3 = 0xaf
res = 0x7791a6af
SubBytes block in and block out:
0x41d7c653, 0x7d669140, 0xdd2f179d, 0x02acc51b
0x830eb4ed, 0xff338109, 0xc115f05e, 0x7791a6af

ShiftRows block in and block out:
0x830eb4ed, 0xff338109, 0xc115f05e, 0x7791a6af
0x8333f0af, 0xff15a6ed, 0xc191b409, 0x770e815e

MixColumns block in and block out:
0x8333f0af, 0xff15a6ed, 0xc191b409, 0x770e815e
0x1741a118, 0x91c99168, 0x8c36386f, 0x23ad82aa

AddRoundKey key, block in and block out:
0xac7766f3, 0x19fadc21, 0x28d12941, 0x575c006e
0x1741a118, 0x91c99168, 0x8c36386f, 0x23ad82aa
0xbb36c7eb, 0x88334d49, 0xa4e7112e, 0x74f182c4

 Final round.
Inside substw:
b0 = 0xbb, b1 = 0x36, b2 = 0xc7, b3 = 0xeb
s0 = 0xea, s1 = 0x05, s2 = 0xc6, s3 = 0xe9
res = 0xea05c6e9
Inside substw:
b0 = 0x88, b1 = 0x33, b2 = 0x4d, b3 = 0x49
s0 = 0xc4, s1 = 0xc3, s2 = 0xe3, s3 = 0x3b
res = 0xc4c3e33b
Inside substw:
b0 = 0xa4, b1 = 0xe7, b2 = 0x11, b3 = 0x2e
s0 = 0x49, s1 = 0x94, s2 = 0x82, s3 = 0x31
res = 0x49948231
Inside substw:
b0 = 0x74, b1 = 0xf1, b2 = 0x82, b3 = 0xc4
s0 = 0x92, s1 = 0xa1, s2 = 0x13, s3 = 0x1c
res = 0x92a1131c
SubBytes block in and block out:
0xbb36c7eb, 0x88334d49, 0xa4e7112e, 0x74f182c4
0xea05c6e9, 0xc4c3e33b, 0x49948231, 0x92a1131c

ShiftRows block in and block out:
0xea05c6e9, 0xc4c3e33b, 0x49948231, 0x92a1131c
0xeac3821c, 0xc49413e9, 0x49a1c63b, 0x9205e331
```

AddRoundKey key, block in and block out:
0xd014f9a8, 0xc9ee2589, 0xe13f0cc8, 0xb6630ca6
0xeac3821c, 0xc49413e9, 0x49a1c63b, 0x9205e331
0x3ad77bb4, 0x0d7a3660, 0xa89ecaf3, 0x2466ef97

OK. Result matches expected.

קודי VERILOG :

//Clk_Mux.v

```
module clk_mux
)#
parameter NUM_OF_CLKS = 8
(
)
input [NUM_OF_CLKS-1:0] i_clk_vec,
input [$clog2(NUM_OF_CLKS)-1:0] i_clk_sel,
output o_clk
;(



.//CLK_SEL_TYPE("ASYNC")  // ASYNC, SYNC
.//INIT_OUT(0), // Initial value of BUFGCTRL output ($VALUES;)
.//PRESELECT_I0("FALSE"), // BUFGCTRL output uses I0 input ($VALUES;)
.//PRESELECT_I1("FALSE") // BUFGCTRL output uses I1 input ($VALUES;)
BUFGMUX)#

(

   BUFGMUX_inst)
.      O(O1),   // 1-bit output: Clock output
.      I0(i_clk_vec[0]), // 1-bit input: Clock input (S=0)
.      I1(i_clk_vec[3]), // 1-bit input: Clock input (S=1)
.      S(i_clk_sel[0])    // 1-bit input: Clock select
;(

.//CLK_SEL_TYPE("ASYNC") // ASYNC, SYNC
.//INIT_OUT(0), // Initial value of BUFGCTRL output ($VALUES;)
.//PRESELECT_I0("FALSE"), // BUFGCTRL output uses I0 input ($VALUES;)
.//PRESELECT_I1("FALSE") // BUFGCTRL output uses I1 input ($VALUES;)
 BUFGMUX)#

(
   BUFGMUX_inst2)
.      O(O2),   // 1-bit output: Clock output
.      I0(O1), // 1-bit input: Clock input (S=0)
.      I1(i_clk_vec[2]), // 1-bit input: Clock input (S=1)
.      S(i_clk_sel[1])    // 1-bit input: Clock select
  ;(
.//CLK_SEL_TYPE("ASYNC")  // ASYNC, SYNC
.//INIT_OUT(0), // Initial value of BUFGCTRL output ($VALUES;)
.//PRESELECT_I0("FALSE"), // BUFGCTRL output uses I0 input ($VALUES;)
.//PRESELECT_I1("FALSE") // BUFGCTRL output uses I1 input ($VALUES;)
  BUFGMUX)#

 (
     BUFGMUX_inst3)
.      O(O3),   // 1-bit output: Clock output
.      I0(O2), // 1-bit input: Clock input (S=0)
.      I1(i_clk_vec[1]), // 1-bit input: Clock input (S=1)
.      S(i_clk_sel[2])    // 1-bit input: Clock select
```

```
;(

.//CLK_SEL_TYPE("ASYNC") // ASYNC, SYNC
.//INIT_OUT(0), // Initial value of BUFGCTRL output ($VALUES;)
.//PRESELECT_I0("FALSE"), // BUFGCTRL output uses I0 input ($VALUES;)
.//PRESELECT_I1("FALSE") // BUFGCTRL output uses I1 input ($VALUES;)
  BUFGMUX)#

 (
        BUFGMUX_inst4)
.        O(o_clk),   // 1-bit output: Clock output
.        I0(i_clk_vec[4]), // 1-bit input: Clock input (S=0)
.        I1(O3), // 1-bit input: Clock input (S=1)
.        S(i_clk_sel[0])    // 1-bit input: Clock select
 ;(
*/


        BUFGMUX)#
(
           BUFGMUX_inst6)
.           O(o_clk),   // 1-bit output: Clock output
.           I0(i_clk_vec[5]), // 1-bit input: Clock input (S=0)
.           I1(O4), // 1-bit input: Clock input (S=1)
.           S(i_clk_sel[1])    // 1-bit input: Clock select
   ;(
/*
endmodule
```

//LFSR_EN.v

```
`timescale 1ns / 1ps

module LFSR_EN

 )#    parameter N = 3)(
    input clk,
    input reset_n,
    output [1:N] Q_en,
    output Enable_out
;(

reg [1:N] Q_reg , Q_next;
wire taps;

reg Q_to_LFSR , En=0;

always @(posedge clk )//, negedge reset_n(
begin

    if (~reset_n)
        Q_reg <= 'd1 ;
    else
        Q_reg <= Q_next;
        Q_to_LFSR <= Q_next;

        if (Q_to_LFSR == 1)
            En <= 1;
        else
            En <= 0;

end

//next stage logic
```

```
always @(taps , Q_reg)
   Q_next <= {taps , Q_reg[1:N-1]};



//output
assign Q_en = Q_reg;

//for N=3
assign taps = Q_reg[3] ^ Q_reg;[2]
assign Enable_out = En;

endmodule
```

// LFSR.v

```
`timescale 1ns / 1ps


module LFSR

 )#    parameter N = 3)(
   input clk,
   input reset_n,
   input Enable,
   output [1:N] Q
;(

reg [1:N] Q_reg , Q_next;
wire taps;

always @(posedge clk , negedge reset_n)
begin

   if (~reset_n)
      Q_reg <= 'd1 ;
    else if (Enable==1)
      Q_reg <= Q_next;

end

//next stage logic

always @(taps , Q_reg)

   Q_next <= {taps , Q_reg[1:N-1]};

//output
assign Q = Q_reg;

//for N=3
assign taps = Q_reg[3] ^ Q_reg;[2]



endmodule
```

//Xor_implimantation

```
//  Truth Table to determine INIT value for a LUT3
//     _______
//     | I2 I1 I0 | O |
//     |--------------|
//     |  0  0  0 | ? |\
//     |  0  0  1 | ? | \ = 4'b???? = 4'h? --------------+
//     |  0  1  0 | ? | /                                |
//     |  0  1  1 | ? |/                                 |
//     |----------|---|                  INIT = 8'h??
//     |  1  0  0 | ? |\                                 |
//     |  1  0  1 | ? | \ = 4'b???? = 4'h? -------------+
//     |  1  1  0 | ? | /
//     |  1  1  1 | ? |/
//     ---------------
module XOR_GATE(

            input A_t,              //true data in #1
            input B_t,              //true data in #2
            output O_t              //true data out
);


wire I2;


assign I2 = 0;


LUT3 #(
.INIT(8'h66) // Specify LUT Contents
) LUT3_inst (
.O(O_t), // LUT general output
.I0(A_t), // LUT input
.I1(B_t), // LUT input
.I2(I2) // LUT input
);
// End of LUT3_inst instantiation

endmodule
```

//MixColumns

```
module MixColumns (
        input       [N-1:0] MixCol_In_T,
        output      [N-1:0] MixCol_Out_T
);
        // module name and ports list.
        //parameters declaration
        parameter N=128; //inputs size
        parameter Nb=4; //number of columns
        parameter BYTE =8; //number of bits in a byte
        parameter WORD = 32; //number of bits in a word

        wire [N-1:0] c_mult_T; //c_mult holds the x_mult output for each byte
        genvar c;

        generate
                for(c=0;c<Nb;c=c+1) begin: xoring

                        xmult mult0(
                                .Input_Byte_T(MixCol_In_T[N-1-c*WORD-:BYTE]),//2*ingf
                                .Output_Byte_T(c_mult_T[N-c*WORD-1-:BYTE])
                        );

                        xmult mult1(
                                .Input_Byte_T(MixCol_In_T[N-c*WORD-BYTE-1-:BYTE]),
                                .Output_Byte_T(c_mult_T[N-c*WORD-1-BYTE-:BYTE])

                        );

                        xmult mult2(
                                .Input_Byte_T(MixCol_In_T[N-c*WORD-2*BYTE-1-:BYTE]),
                                .Output_Byte_T(c_mult_T[N-c*WORD-1-2*BYTE-:BYTE])
                        );

                        xmult mult3(
                                .Input_Byte_T(MixCol_In_T[N-c*WORD-3*BYTE-1-:BYTE]),
                                .Output_Byte_T(c_mult_T[N-c*WORD-1-3*BYTE-:BYTE])

                        );


                        XOR5_BYTE Xor1(
                                .In1_T(c_mult_T[N-c*WORD-1-:BYTE]),
                                .In2_T(c_mult_T[N-c*WORD-1-BYTE-:BYTE]),
                                .In3_T(MixCol_In_T[N-c*WORD-BYTE-1-:BYTE]),
                                .In4_T(MixCol_In_T[N-c*WORD-2*BYTE-1-:BYTE]),
                                .In5_T(MixCol_In_T[N-c*WORD-3*BYTE-1-:BYTE]),
                                .Out_T(MixCol_Out_T[N-1-c*WORD-:BYTE])
                        );


                        XOR5_BYTE Xor2(
                                .In1_T(MixCol_In_T[N-1-c*WORD-:BYTE]),
                                .In2_T(c_mult_T[N-c*WORD-1-BYTE-:BYTE]),
                                .In3_T(c_mult_T[N-c*WORD-1-2*BYTE-:BYTE]),
                                .In4_T(MixCol_In_T[N-c*WORD-2*BYTE-1-:BYTE]),
                                .In5_T(MixCol_In_T[N-c*WORD-3*BYTE-1-:BYTE]),
                                .Out_T(MixCol_Out_T[N-c*WORD-BYTE-1-:BYTE])
                        );
```

```
                XOR5_BYTE Xor3(
                        .In1_T(MixCol_In_T[N-1-c*WORD-:BYTE]),
                        .In2_T(MixCol_In_T[N-c*WORD-BYTE-1-:BYTE]),
                        .In3_T(c_mult_T[N-c*WORD-1-2*BYTE-:BYTE]),
                        .In4_T(c_mult_T[N-c*WORD-1-3*BYTE-:BYTE]),
                        .In5_T(MixCol_In_T[N-c*WORD-3*BYTE-1-:BYTE]),
                        .Out_T(MixCol_Out_T[N-c*WORD-2*BYTE-1-:BYTE])
                );



                XOR5_BYTE Xor4(
                        .In1_T(c_mult_T[N-c*WORD-1-:BYTE]),
                        .In2_T(MixCol_In_T[N-1-c*WORD-:BYTE]),
                        .In3_T(MixCol_In_T[N-c*WORD-BYTE-1-:BYTE]),
                        .In4_T(MixCol_In_T[N-c*WORD-2*BYTE-1-:BYTE]),
                        .In5_T(c_mult_T[N-c*WORD-1-3*BYTE-:BYTE]),
                        .Out_T(MixCol_Out_T[N-c*WORD-3*BYTE-1-:BYTE])
                );


                end
        endgenerate

endmodule

module xmult (
        input   [BYTE-1:0] Input_Byte_T,
        output  [BYTE-1:0] Output_Byte_T
);
        // checks the MSB of the Input_Byte: if it is 0 ולא shift left
        // if it is a 1 ולא shift to the left and XOR with {1B}.

        parameter BYTE = 8;
        wire tmpXOR1_T,tmpXOR3_T,tmpXOR4_T;


        XOR_GATE XOR_1(
                .A_t(Input_Byte_T[0]),
                .B_t(1'b1),
                .O_t(tmpXOR1_T)
        );

        XOR_GATE XOR_2(
                .A_t(Input_Byte_T[2]),
                .B_t(1'b1),
                .O_t(tmpXOR3_T)
        );

        XOR_GATE XOR_3(
                .A_t(Input_Byte_T[3]),
                .B_t(1'b1),
                .O_t(tmpXOR4_T)
        );
        assign Output_Byte_T[0] = (!Input_Byte_T[7]) ? 1'b0 : 1'b1;
        assign Output_Byte_T[1] = (!Input_Byte_T[7]) ? Input_Byte_T[0] : tmpXOR1_T;
        assign Output_Byte_T[2] = Input_Byte_T[1];
        assign Output_Byte_T[3] = (!Input_Byte_T[7]) ? Input_Byte_T[2] : tmpXOR3_T;
        assign Output_Byte_T[4] = (!Input_Byte_T[7]) ? Input_Byte_T[3] : tmpXOR4_T;
        assign Output_Byte_T[7:5] = Input_Byte_T[6:4];

endmodule
```

//ShiftRows

```
module ShiftRows (
        input       [N-1:0]     Text_In_T,
        output      [N-1:0]     Out_Text_T
);
        // module name and ports list.
        //parameters declaration
        parameter N=128; //inputs size
        parameter BYTE =8; //number of bits in a byte
        parameter WORD = 32; //number of bits in a word

        //TRUE
        //first row
        assign Out_Text_T[WORD-1:0] = {Text_In_T[(WORD-1):(WORD-BYTE)],Text_In_T[(N-BYTE-1):(N-
2*BYTE)],Text_In_T[(2*WORD+2*BYTE-1):(2*WORD+BYTE)],

        Text_In_T[(WORD+BYTE-1):(WORD)]};
        //second row
        assign Out_Text_T[2*WORD-1-:WORD] = {Text_In_T[(2*WORD-1)-:BYTE],Text_In_T[(3*BYTE-1)-
:BYTE],Text_In_T[(3*WORD+2*BYTE-1)-:BYTE],
                                                Text_In_T[(2*WORD+BYTE-1)-:BYTE]};
        //third row
        assign Out_Text_T[3*WORD-1-:WORD] = {Text_In_T[(3*WORD-1)-:BYTE],Text_In_T[(2*WORD-BYTE-1)-
:BYTE],Text_In_T[(2*BYTE-1)-:BYTE],
                                                Text_In_T[(3*WORD+BYTE-1)-:BYTE]};
        //forth word
        assign Out_Text_T[4*WORD-1-:WORD] = {Text_In_T[(4*WORD-1)-:BYTE],Text_In_T[(3*WORD-BYTE-1)-
:BYTE],Text_In_T[(2*WORD-2*BYTE-1)-:BYTE],
                                                Text_In_T[(BYTE-1)-:BYTE]};

endmodule
```

//AddRoundKey.v

```
module AddRoundKey (                    // module name and ports list.
        input               [N-1:0]     In_Data,
        input               [N-1:0]     In_key,
        output reg [N-1:0]      Output_Key
);
        //parameters declaration
        parameter N=128; //input size

        always@(*)
                Output_Key = In_Data ^ In_key; //bitwise XOR

endmodule
```

//SubBytes

```
module SubBytes (
        input                   Clk,
        input   [1:0]   Multi_Cycle_State,
        input   [N-1:0]  SubByte_In_T,
        output  [N-1:0]  SubByte_Out_T
);              // module name and ports list.

        parameter N=128; //inputs size
        parameter BYTE =8; //number of bits in a byte

        genvar i;

        generate
        for (i=0; i< N/BYTE; i=i+1)begin: eightbit
                Sbox8b sbox (
                        .Clk(Clk),
                        .multi_cycle(Multi_Cycle_State),
                        .Sbox_In_T(SubByte_In_T[i*BYTE+:BYTE]),
                        .Sbox_Out_T(SubByte_Out_T[i*BYTE+:BYTE])
                );


        end
        endgenerate

endmodule
```

//FSM_AES

```
module Regular_AES256_FSMSboxState (
        input                           Clk,
        input           [N-1:0]  Plain_text,
        input                           Reset,
        input           [K-1:0]  In_Key,
        output reg [N-1:0]  Cipher_text,
        output reg              Done,
        output reg              trigger
);

        parameter N = 128;
        parameter K = 256;
        parameter ROUND = 4;


        //wires
        wire [K-1:0]            Next_Key_T ;

        wire [N-1:0]            Sliced_Key_T;
        wire [N-1:0]            RoundKey_Out_wire_T;
        //reg [N-1:0]           RoundKey_Out_wire_T_temp;
        wire [N-1:0]            Sbox_Out_wire_T;
        wire [N-1:0]            Shift_Rows_wire_out_T;
        wire [N-1:0]            Mix_Col_wire_In_T;
        wire [N-1:0]            Mix_Col_wire_out_T;

        //registers
        reg [K-1:0] Current_Key_T;

        wire [N-1:0]            Sbox_In_wire_T;
```

```
        reg [N-1:0]             State_reg_T;
        reg [N-1:0]             AddRoundIn_T;
        reg [ROUND-1:0] Round_Wire;
        reg [2:0]               Multi_State_Ex;
        reg [1:0]               State, State_Wire;
        reg [1:0]               Multi_Cycle_Wire;
        reg [ROUND-1:0] Round;
        reg                                     Done_wire;
reg         Done_temp;
reg [ROUND-1:0]  RoundEx_in_reg_T_temp;
 reg [1:0]     State_temp;
        reg [ROUND-1:0] Round_temp;
                reg [N-1:0]             State_reg_T_temp;
        reg [K-1:0] Current_Key_T_temp;

                wire [ROUND-1:0]        RoundEx_in_wire_T ;
                reg [ROUND-1:0]         RoundEx_in_reg_T ;


        assign RoundEx_in_wire_T = (Reset) ? 4'h0 : (Round_Wire[0] == 1'b1) ? (Round>>1) : RoundEx_in_reg_T_temp;

        assign Sliced_Key_T = (Round[0] == 1'b1) ? Current_Key_T_temp[N-1:0] : Current_Key_T_temp[K-1:N];

        assign Mix_Col_wire_In_T = Shift_Rows_wire_out_T;

        assign Sbox_In_wire_T = RoundKey_Out_wire_T;


        KeyExpantion_256 KeyEx(
                .Clk(Clk),
                .Multi_State(Multi_State_Ex),
                .In_Key_T(Current_Key_T),
                .Round_Number_T(RoundEx_in_wire_T),
                .Out_Key_T(Next_Key_T)
        );

        AddRoundKey key(
                .In_Data(AddRoundIn_T),
                .In_key(Sliced_Key_T),
                .Output_Key(RoundKey_Out_wire_T)
        );

        SubBytes Sub(
                .Clk(Clk),
                .Multi_Cycle_State(Multi_Cycle_Wire),
                .SubByte_In_T(Sbox_In_wire_T),
                .SubByte_Out_T(Sbox_Out_wire_T)
        );

        ShiftRows shft(
                .Text_In_T(State_reg_T),
                .Out_Text_T(Shift_Rows_wire_out_T)
        );

        MixColumns MxCl(
                .MixCol_In_T(Mix_Col_wire_In_T),
                .MixCol_Out_T(Mix_Col_wire_out_T)
        );



        always @(*)
        begin
```

```
case (State)

2'd0:begin
        Round_Wire = 0;
        State_Wire = 2'd1;
        Multi_Cycle_Wire = 2'd0;
        Cipher_text = 0;
        Multi_State_Ex = 3'd0;
        Done_wire = 0;
        trigger = 0;
        AddRoundIn_T = 0;

        end

2'd1:begin

        Multi_Cycle_Wire = 2'd1;
        Cipher_text = 0;
        Done_wire = 1'd0;
        Round_Wire = Round;
        trigger = 0;

        if (Round == 0) begin
                AddRoundIn_T = Plain_text;
        end
        else begin
                AddRoundIn_T = Mix_Col_wire_out_T;
        end

        if (Round[0] == 0) begin
                Multi_State_Ex = 3'd1;
        end
        else begin
                Multi_State_Ex = 3'd4;

        end
        if (Round == 4'd14) begin
                Cipher_text = Shift_Rows_wire_out_T ^ Sliced_Key_T;
                State_Wire = 2'd0;
        end
        else begin
                State_Wire = 2'd2;
                Cipher_text = 0;
        end

        end

2'd2:begin
        Multi_Cycle_Wire = 2'd2;
        State_Wire = 2'd3;
        Cipher_text = 0;
        AddRoundIn_T = Mix_Col_wire_out_T;
        Done_wire = 0;
        Round_Wire = Round;

        if (Round[0] == 0) begin
                Multi_State_Ex = 3'd2;
        end
        else begin
                Multi_State_Ex = 3'd5;

        end


        if (Round == 4'd2)
                trigger = 1'd1;
        else
```

```
                        trigger = 0;


                end

        2'd3:begin
                Multi_Cycle_Wire = 2'd3;
                AddRoundIn_T = Mix_Col_wire_out_T;
                State_Wire = 2'd1;
                Round_Wire = Round + 4'd1;
                Cipher_text = 0;
                trigger = 0;


                if (Round[0] == 0) begin
                        Multi_State_Ex = 3'd3;

                end
                else begin
                        Multi_State_Ex = 3'd6;

                end

                if (Round == 4'd13)
                        Done_wire = 1'd1;
                else
                        Done_wire = 0;

                end

        endcase
        end


        always @(posedge Clk)
        begin
                Done_temp <= Done_wire;
                RoundEx_in_reg_T_temp <= RoundEx_in_wire_T;

                if (Reset) begin
                        State_temp <= 0;
                        Round_temp <= 0;
                        State_reg_T_temp <= 0;
                        Current_Key_T_temp <= In_Key;

                end
                else begin
                  // RoundEx_in_reg_T_temp <= RoundEx_in_wire_T;
                        State_temp <= State_Wire;
                        //RoundKey_Out_wire_T_temp<=RoundKey_Out_wire_T;
                        Round_temp <= Round_Wire;

                        State_reg_T_temp <= Sbox_Out_wire_T;

                        if ((Round[0] == 1'd1) && (State == 2'd3)) begin
                                Current_Key_T_temp <= Next_Key_T;
                        end

                end

        end

always @(posedge Clk)
        begin
        if (Reset) begin
                        State <= 0;
                        Round <= 0;
```

```
                                State_reg_T <= 0;
                                Current_Key_T <= In_Key;
                                Done<=0;

                        end
                        else begin
//              if Done!=1 begin
        Done <= Done_temp;
        RoundEx_in_reg_T <= RoundEx_in_reg_T_temp;
        State <= State_temp;
        Round <= Round_temp;
        State_reg_T <= State_reg_T_temp;
        Current_Key_T <= Current_Key_T_temp;
    end
                end

endmodule
```

// The sbox array.v

```
wire [7 : 0] sbox [0 : 255];


//-----------------------------------------------------------------
// Four parallel muxes.
//-----------------------------------------------------------------
assign new_sboxw[31 : 24] = sbox[sboxw[31 : 24]];
assign new_sboxw[23 : 16] = sbox[sboxw[23 : 16]];
assign new_sboxw[15 : 08] = sbox[sboxw[15 : 08]];
assign new_sboxw[07 : 00] = sbox[sboxw[07 : 00]];


//-----------------------------------------------------------------
// Creating the sbox array contents.
//-----------------------------------------------------------------
assign sbox[8'h00] = 8'h63;
assign sbox[8'h01] = 8'h7c;
assign sbox[8'h02] = 8'h77;
assign sbox[8'h03] = 8'h7b;
assign sbox[8'h04] = 8'hf2;
assign sbox[8'h05] = 8'h6b;
assign sbox[8'h06] = 8'h6f;
assign sbox[8'h07] = 8'hc5;
assign sbox[8'h08] = 8'h30;
assign sbox[8'h09] = 8'h01;
assign sbox[8'h0a] = 8'h67;
assign sbox[8'h0b] = 8'h2b;
assign sbox[8'h0c] = 8'hfe;
assign sbox[8'h0d] = 8'hd7;
assign sbox[8'h0e] = 8'hab;
assign sbox[8'h0f] = 8'h76;
assign sbox[8'h10] = 8'hca;
assign sbox[8'h11] = 8'h82;
assign sbox[8'h12] = 8'hc9;
assign sbox[8'h13] = 8'h7d;
assign sbox[8'h14] = 8'hfa;
assign sbox[8'h15] = 8'h59;
assign sbox[8'h16] = 8'h47;
assign sbox[8'h17] = 8'hf0;
assign sbox[8'h18] = 8'had;
assign sbox[8'h19] = 8'hd4;
assign sbox[8'h1a] = 8'ha2;
assign sbox[8'h1b] = 8'haf;
assign sbox[8'h1c] = 8'h9c;
assign sbox[8'h1d] = 8'ha4;
assign sbox[8'h1e] = 8'h72;
assign sbox[8'h1f] = 8'hc0;
assign sbox[8'h20] = 8'hb7;
assign sbox[8'h21] = 8'hfd;
assign sbox[8'h22] = 8'h93;
assign sbox[8'h23] = 8'h26;
assign sbox[8'h24] = 8'h36;
assign sbox[8'h25] = 8'h3f;
assign sbox[8'h26] = 8'hf7;
assign sbox[8'h27] = 8'hcc;
assign sbox[8'h28] = 8'h34;
assign sbox[8'h29] = 8'ha5;
assign sbox[8'h2a] = 8'he5;
assign sbox[8'h2b] = 8'hf1;
assign sbox[8'h2c] = 8'h71;
assign sbox[8'h2d] = 8'hd8;
assign sbox[8'h2e] = 8'h31;
```

```
assign sbox[8'h2f] = 8'h15;
assign sbox[8'h30] = 8'h04;
assign sbox[8'h31] = 8'hc7;
assign sbox[8'h32] = 8'h23;
assign sbox[8'h33] = 8'hc3;
assign sbox[8'h34] = 8'h18;
assign sbox[8'h35] = 8'h96;
assign sbox[8'h36] = 8'h05;
assign sbox[8'h37] = 8'h9a;
assign sbox[8'h38] = 8'h07;
assign sbox[8'h39] = 8'h12;
assign sbox[8'h3a] = 8'h80;
assign sbox[8'h3b] = 8'he2;
assign sbox[8'h3c] = 8'heb;
assign sbox[8'h3d] = 8'h27;
assign sbox[8'h3e] = 8'hb2;
assign sbox[8'h3f] = 8'h75;
assign sbox[8'h40] = 8'h09;
assign sbox[8'h41] = 8'h83;
assign sbox[8'h42] = 8'h2c;
assign sbox[8'h43] = 8'h1a;
assign sbox[8'h44] = 8'h1b;
assign sbox[8'h45] = 8'h6e;
assign sbox[8'h46] = 8'h5a;
assign sbox[8'h47] = 8'ha0;
assign sbox[8'h48] = 8'h52;
assign sbox[8'h49] = 8'h3b;
assign sbox[8'h4a] = 8'hd6;
assign sbox[8'h4b] = 8'hb3;
assign sbox[8'h4c] = 8'h29;
assign sbox[8'h4d] = 8'he3;
assign sbox[8'h4e] = 8'h2f;
assign sbox[8'h4f] = 8'h84;
assign sbox[8'h50] = 8'h53;
assign sbox[8'h51] = 8'hd1;
assign sbox[8'h52] = 8'h00;
assign sbox[8'h53] = 8'hed;
assign sbox[8'h54] = 8'h20;
assign sbox[8'h55] = 8'hfc;
assign sbox[8'h56] = 8'hb1;
assign sbox[8'h57] = 8'h5b;
assign sbox[8'h58] = 8'h6a;
assign sbox[8'h59] = 8'hcb;
assign sbox[8'h5a] = 8'hbe;
assign sbox[8'h5b] = 8'h39;
assign sbox[8'h5c] = 8'h4a;
assign sbox[8'h5d] = 8'h4c;
assign sbox[8'h5e] = 8'h58;
assign sbox[8'h5f] = 8'hcf;
assign sbox[8'h60] = 8'hd0;
assign sbox[8'h61] = 8'hef;
assign sbox[8'h62] = 8'haa;
assign sbox[8'h63] = 8'hfb;
assign sbox[8'h64] = 8'h43;
assign sbox[8'h65] = 8'h4d;
assign sbox[8'h66] = 8'h33;
assign sbox[8'h67] = 8'h85;
assign sbox[8'h68] = 8'h45;
assign sbox[8'h69] = 8'hf9;
assign sbox[8'h6a] = 8'h02;
assign sbox[8'h6b] = 8'h7f;
assign sbox[8'h6c] = 8'h50;
assign sbox[8'h6d] = 8'h3c;
assign sbox[8'h6e] = 8'h9f;
assign sbox[8'h6f] = 8'ha8;
assign sbox[8'h70] = 8'h51;
```

```
assign sbox[8'h71] = 8'ha3;
assign sbox[8'h72] = 8'h40;
assign sbox[8'h73] = 8'h8f;
assign sbox[8'h74] = 8'h92;
assign sbox[8'h75] = 8'h9d;
assign sbox[8'h76] = 8'h38;
assign sbox[8'h77] = 8'hf5;
assign sbox[8'h78] = 8'hbc;
assign sbox[8'h79] = 8'hb6;
assign sbox[8'h7a] = 8'hda;
assign sbox[8'h7b] = 8'h21;
assign sbox[8'h7c] = 8'h10;
assign sbox[8'h7d] = 8'hff;
assign sbox[8'h7e] = 8'hf3;
assign sbox[8'h7f] = 8'hd2;
assign sbox[8'h80] = 8'hcd;
assign sbox[8'h81] = 8'h0c;
assign sbox[8'h82] = 8'h13;
assign sbox[8'h83] = 8'hec;
assign sbox[8'h84] = 8'h5f;
assign sbox[8'h85] = 8'h97;
assign sbox[8'h86] = 8'h44;
assign sbox[8'h87] = 8'h17;
assign sbox[8'h88] = 8'hc4;
assign sbox[8'h89] = 8'ha7;
assign sbox[8'h8a] = 8'h7e;
assign sbox[8'h8b] = 8'h3d;
assign sbox[8'h8c] = 8'h64;
assign sbox[8'h8d] = 8'h5d;
assign sbox[8'h8e] = 8'h19;
assign sbox[8'h8f] = 8'h73;
assign sbox[8'h90] = 8'h60;
assign sbox[8'h91] = 8'h81;
assign sbox[8'h92] = 8'h4f;
assign sbox[8'h93] = 8'hdc;
assign sbox[8'h94] = 8'h22;
assign sbox[8'h95] = 8'h2a;
assign sbox[8'h96] = 8'h90;
assign sbox[8'h97] = 8'h88;
assign sbox[8'h98] = 8'h46;
assign sbox[8'h99] = 8'hee;
assign sbox[8'h9a] = 8'hb8;
assign sbox[8'h9b] = 8'h14;
assign sbox[8'h9c] = 8'hde;
assign sbox[8'h9d] = 8'h5e;
assign sbox[8'h9e] = 8'h0b;
assign sbox[8'h9f] = 8'hdb;
assign sbox[8'ha0] = 8'he0;
assign sbox[8'ha1] = 8'h32;
assign sbox[8'ha2] = 8'h3a;
assign sbox[8'ha3] = 8'h0a;
assign sbox[8'ha4] = 8'h49;
assign sbox[8'ha5] = 8'h06;
assign sbox[8'ha6] = 8'h24;
assign sbox[8'ha7] = 8'h5c;
assign sbox[8'ha8] = 8'hc2;
assign sbox[8'ha9] = 8'hd3;
assign sbox[8'haa] = 8'hac;
assign sbox[8'hab] = 8'h62;
assign sbox[8'hac] = 8'h91;
assign sbox[8'had] = 8'h95;
assign sbox[8'hae] = 8'he4;
assign sbox[8'haf] = 8'h79;
assign sbox[8'hb0] = 8'he7;
assign sbox[8'hb1] = 8'hc8;
assign sbox[8'hb2] = 8'h37;
```

```
assign sbox[8'hb3] = 8'h6d;
assign sbox[8'hb4] = 8'h8d;
assign sbox[8'hb5] = 8'hd5;
assign sbox[8'hb6] = 8'h4e;
assign sbox[8'hb7] = 8'ha9;
assign sbox[8'hb8] = 8'h6c;
assign sbox[8'hb9] = 8'h56;
assign sbox[8'hba] = 8'hf4;
assign sbox[8'hbb] = 8'hea;
assign sbox[8'hbc] = 8'h65;
assign sbox[8'hbd] = 8'h7a;
assign sbox[8'hbe] = 8'hae;
assign sbox[8'hbf] = 8'h08;
assign sbox[8'hc0] = 8'hba;
assign sbox[8'hc1] = 8'h78;
assign sbox[8'hc2] = 8'h25;
assign sbox[8'hc3] = 8'h2e;
assign sbox[8'hc4] = 8'h1c;
assign sbox[8'hc5] = 8'ha6;
assign sbox[8'hc6] = 8'hb4;
assign sbox[8'hc7] = 8'hc6;
assign sbox[8'hc8] = 8'he8;
assign sbox[8'hc9] = 8'hdd;
assign sbox[8'hca] = 8'h74;
assign sbox[8'hcb] = 8'h1f;
assign sbox[8'hcc] = 8'h4b;
assign sbox[8'hcd] = 8'hbd;
assign sbox[8'hce] = 8'h8b;
assign sbox[8'hcf] = 8'h8a;
assign sbox[8'hd0] = 8'h70;
assign sbox[8'hd1] = 8'h3e;
assign sbox[8'hd2] = 8'hb5;
assign sbox[8'hd3] = 8'h66;
assign sbox[8'hd4] = 8'h48;
assign sbox[8'hd5] = 8'h03;
assign sbox[8'hd6] = 8'hf6;
assign sbox[8'hd7] = 8'h0e;
assign sbox[8'hd8] = 8'h61;
assign sbox[8'hd9] = 8'h35;
assign sbox[8'hda] = 8'h57;
assign sbox[8'hdb] = 8'hb9;
assign sbox[8'hdc] = 8'h86;
assign sbox[8'hdd] = 8'hc1;
assign sbox[8'hde] = 8'h1d;
assign sbox[8'hdf] = 8'h9e;
assign sbox[8'he0] = 8'he1;
assign sbox[8'he1] = 8'hf8;
assign sbox[8'he2] = 8'h98;
assign sbox[8'he3] = 8'h11;
assign sbox[8'he4] = 8'h69;
assign sbox[8'he5] = 8'hd9;
assign sbox[8'he6] = 8'h8e;
assign sbox[8'he7] = 8'h94;
assign sbox[8'he8] = 8'h9b;
assign sbox[8'he9] = 8'h1e;
assign sbox[8'hea] = 8'h87;
assign sbox[8'heb] = 8'he9;
assign sbox[8'hec] = 8'hce;
assign sbox[8'hed] = 8'h55;
assign sbox[8'hee] = 8'h28;
assign sbox[8'hef] = 8'hdf;
assign sbox[8'hf0] = 8'h8c;
assign sbox[8'hf1] = 8'ha1;
assign sbox[8'hf2] = 8'h89;
assign sbox[8'hf3] = 8'h0d;
assign sbox[8'hf4] = 8'hbf;
```

```
  assign sbox[8'hf5] = 8'he6;
  assign sbox[8'hf6] = 8'h42;
  assign sbox[8'hf7] = 8'h68;
  assign sbox[8'hf8] = 8'h41;
  assign sbox[8'hf9] = 8'h99;
  assign sbox[8'hfa] = 8'h2d;
  assign sbox[8'hfb] = 8'h0f;
  assign sbox[8'hfc] = 8'hb0;
  assign sbox[8'hfd] = 8'h54;
  assign sbox[8'hfe] = 8'hbb;
  assign sbox[8'hff] = 8'h16;

endmodule // aes_sbox
```

Python

קלט המדידות, הורדת הרעש שהטריגר מייצר, הורדת DC, ומצוי כלל המדידות ושמרתם בקובץ mat.

```
from scipy import signal
import csv
import matplotlib.pyplot as plt
import sys
import wave
import numpy as np
import time
import matplotlib.pyplot as plt
import scipy.signal as sci_sig
import librosa as lib
import soundfile
import sys
import  os
import librosa, librosa.display
from scipy.stats import laplace
import plotly.graph_objects as go
import datetime
import re
import pandas as pd
def removedc(datanow):
    """
Args::  file lowers dc frequency
Returns: the file without DC
"""
    N = datanow.shape[0]
    datafft=np.fft.fft(datanow)/N
    plt.show()
    datafft[0]=0
    plt.show()
    datanow=np.fft.irfft(datafft,N)
    return  datanow
indix=333
unitlist=[]
dictall={}
files=[]
count=0
# Input data into an array with the file name and its data
files=os.listdir('C:\\Users\\spring1\\Documents\\MATLAB\\AES\\TXT')
files=[i for i in files if re.findall(".+.txt", i )]
for file in files:
    cfile=os.path.join('C:\\Users\\spring1\\Documents\\MATLAB\\AES\\TXT', file)
    print(file)#check
    unitlist=[]


    f = open(cfile)
    triplets=f.read().split()
    for i in range(0,len(triplets)): triplets[i]=triplets[i].split(',')
    j=0
    s=0
    zeors1= np.zeros( (380, 65280) )
    for i in triplets:

        zeors1[j,s]=i[0]
        s=s+1
```

```
        if s==65280:
            j+=1
            s=0
            print(j)#check
    narray=zeors1.T
    list_remove=set()
    remoeall=False
    ofsset=0
    narray1=narray[0,:]
    narray2=narray
    for i in range(0,narray.shape[0]):
        narray2[i,:]=removedc(narray[i,:])
# Check the desired voltage level for lowering (all the voltage above it)
# (so that the voltage generated by the trigger does not affect the properties)
    if count==0:
        print('aaaaaaaaa') #check
        for i,v in np.ndenumerate(narray1):

            if abs(v)>0.017 or remoeall:

                indix=i[0]
                count+=1
                break

    for i in range(indix,narray.shape[1]):
        narray2=np.delete(narray2,i-ofsset,1)
        ofsset+=1
    dictall[file] =   narray2
# Averaging all the data obtained to create an extraction array between all the data
zeors1= np.zeros( (narray2.shape[0], narray2.shape[1]) )
for i in dictall.values():
    zeors1+=i
aver=(zeors1/len(dictall))
from scipy.io import savemat

mdic = {"P__AESRANDOM_P1__": aver, "label": "experiment"}


savemat("P__AESRANDOM_P1__.mat", mdic)
```

Matlab:

יצרת מטריצת X(מטריצת כלל plain text),בKEY כותב את המפתח הסודי שמשתמשים בו ,בin כותבים את הplain text (data)הראשוני שרוצים להצפין

```
key='00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff';%The key that
users use
In='00000000000000000000000000000000';%plain text(The original unencrypted information
that enters the encryption algorithm)
A=[];
% A(i,:)=In;
D=[];
n=1;
% Creating the plain text rule (for the number of encryption rounds performed)
% (each time the output becomes the new plain text)And insert it into matrix X (plain
text matrix)
for i =1:65280
% Revenue Revenue 2 to 65280
if (i~=1)
Out = Cipher(key,In);%encryption(AES) function of matlab
In=Out;
end
%Revenue Revenue 1
if (i==1)
Out='00000000000000000000000000000000' ;
end
n=1;

for j =1:2:32
D(n)=hex2dec( Out(j:1+j));%Conversion from base hex to dec
n=n+1;
end
A(i,:)=D;
end
% save('X11.mat','A');%X(PLAIN TEXT) Save the matrix
```

יצרית מטריצות XxorK וB כמו שכתוב בפרק מערך התקיפה

```
size_of_aes =128;
bytes_of_AES = size_of_aes/8;
AES_key_bit = 2^8;
num_of_enc = 65280;
example = matfile('X1.mat');%Information claim
X = example.A;
XxorK=Xxork(num_of_enc,bytes_of_AES,AES_key_bit,X);
B=b(num_of_enc,bytes_of_AES,AES_key_bit,XxorK);
% save('XxorK111.mat','XxorK');%Save the matrix
% save('B_MAT11.mat','B');%Save the matrix
```

פונקציות המייצרות את היפותזות של המפתחות בשיטת HD,HW כמו שכתוב בפרק מערך התקיפה

```
%% HW( Function creating a hamming weight matrix)
load('XxorK1.mat');%Information claim
load('B_MAT1.mat');%Information claim
HW_B=HW(B,bytes_of_AES,AES_key_bit,num_of_enc);
% save('HW_B_AES256_Cipher3.mat','HW_B');%Save the matrix
%% HD ( Function creating a hamming distance matrix)
load('XxorK1.mat');%Information claim
load('B_MAT1.mat');%Information claim
HD_B=HD(B,bytes_of_AES,AES_key_bit,num_of_enc,XxorK);
% HD1_B=HD1(B,bytes_of_AES,AES_key_bit,num_of_enc);
% save('HD_B_AES256_Cipher3.mat','HD_B');%Save the matrix
```

בדיקת CR,correlation בין מטריצות ההיפותזות למטריצה p (מטריצת המדידות) כמו שכתוב בפרק מערך התקיפה

```
num_of_The_exchangesbox = 16;
size_of_aes =128;
bytes_of_AES = size_of_aes/8;
AES_key_bit = 2^8;
num_of_enc = 65280;
%gather-Transfers code execution to gpu
num_of_sbox = 16;
keys_per_sbox = 256;
num_inputs = 65280;
crhd=gpuArray([]);
crhw=gpuArray([]);
correlation_HW=[];
correlation_HD=[];
CR_HD=[];
CR_HW=[];
crhd1=[];
crhw1=[];
hwcrall=[];
hdcrall=[];
% load('B_HW_AES256.mat');%Information claim
% load('B_HD_AES256.mat');%Information claim
for num_inputs=1:1:65280
num_inputs
% tic
init_key = '00112233445566778899aabbccddeeff';
% load('Regular_AES256.mat');
% Imat = I_save';
% ssamples_P = size(Imat,2);
samples_P=333;
load('P__AESRANDOM_P1_.mat');%Information claim P matrix(aes random
clk)
Imat=P__AESRANDOM_P1_;
Imat1 = gpuArray(zeros(num_inputs,samples_P));
for j=1:num_inputs
    Imat1(j,:)=gpuArray(Imat(j,:));
end
Imat = gpuArray(Imat1);
for SboxNum=1:num_of_sbox
    correlation_HW(:,:,SboxNum) =
corr(gpuArray(Imat),gpuArray(HW_B(1:num_inputs,:,SboxNum)));
    correlation_HD(:,:,SboxNum) =
corr(Imat,HD_B(1:num_inputs,:,SboxNum));
    correlation_HD1=gather(correlation_HD);
    correlation_HW1=gather(correlation_HW);
    correlation_HW=gpuArray(correlation_HW);
    correlation_HD=gpuArray(correlation_HD);
%correlation ratio (CR)
%CR - The ratio between the correlation of the correct hypothesis
(max correlation)
%and the first magnitude incorrect correlation (second max
correlation).
    current_key = gpuArray(hex2dec(init_key(2*SboxNum - 1:2*SboxNum)))+1;
     CR_HD(SboxNum) = CR(correlation_HD1(:,:,SboxNum),(current_key));
     CR_HW(SboxNum) = CR(correlation_HW1(:,:,SboxNum),((current_key)));
```

```
end
crhd1(num_inputs)=(CR_HD(1));
crhw1(num_inputs)=(CR_HW(1));
for key=1:16
    hwcrall(:,num_inputs,key)=CR_HW(key);
    hdcrall(:,num_inputs,key)=CR_HD(key);

end
% toc
End
```

PLOT:

```
%% PLOT correlation
samples = 0:1:(samples_P-1);
sampling_rate = 2.5e9;
time = ((1/sampling_rate)*samples) *1e9;
figure
 N=1;
 hold on
 for ii=1:keys_per_sbox
    if(ii~=100)
        plot(time,correlation_HW(:,ii,1), 'b')
    end
     if (ii==1)
        plot(time,correlation_HW(:,1,1),     'r')

     end
end
plot(time,correlation_HW(:,1,1), 'r')

legend({'Hypothesis Correlation', 'Key Correlation'}, 'FontSize', 12)
title(' AES256 RANDOM CLK (HW)', 'FontSize', 20)
% title(' Regular AES256R   (HW)', 'FontSize', 20)
xlabel('TIME[ns]', 'FontSize', 12)
ylabel('Correlation', 'FontSize', 12)
%% PLOT CR
%%
figure
hold on
for key=1:16


nabla(key)=key;

plot(1:1000:65001,hdcrall(:,(1:1000:65001),key));
if key==16
yl = yline(1,'-','correlation ratio>1 attack was successfull(Continues to be
higher than 1)','LineWidth',4);
end


% lgd=legend({' red line =1 (The height=1)', 'correlation ratio key1
'},'Location','east', 'FontSize', 10);
%
end
```

```
leg=string(nabla); %converts integer-array to string array
 for j = 1:17
   if (j~=17)
   leg(j) = 'key ' + leg(j);
   end
   if (j==17)
   leg(j) = ' above red line=correlation ratio>1 ';
   legend(leg)
   end
 end

% lgd=legend({' red line =1 (The height=1)', 'correlation ratio key{key}
'},'Location','east', 'FontSize', 10);

% title('Regular AES256(HD)', 'FontSize', 20)
%
title(' AES256 RANDOM CLK (HD)', 'FontSize', 20)
xlabel('number of traces', 'FontSize', 12)
ylabel('correlation ratio', 'FontSize', 12)
xlim([0 6.6E4]);
% ylim([0 1.2]);
% hold on
% yl = yline(1,'-','correlation ratio>1 attack was successfull(Continues to
be higher than 1)','LineWidth',4);
yl.LabelHorizontalAlignment = 'center';
yl.Color = [.80 0 .40];
ylim([0 1.2]);
% legend({' red line =1 (The height=1)', 'correlation ratio'}, 'FontSize',
10.7)

%%
figure
% title('Regular AES256(HW)', 'FontSize', 20)
%
title(' AES256 RANDOM CLK (HW)', 'FontSize', 20)
xlabel('number of traces', 'FontSize', 12)
ylabel('correlation ratio', 'FontSize', 12)
xlim([0 6.6E4]);
ylim([0 1.2]);
hold on
yl = yline(1,'-','correlation ratio>1 attack was successfull(Continues to be
higher than 1)','LineWidth',4);
yl.LabelHorizontalAlignment = 'center';
yl.Color = [.80 0 .40];
plot(1:100:65280,crcheck_hw4(:,1:100:65280))
legend({' above red line=correlation ratio>1 ', 'correlation ratio'},
'FontSize', 10.7)
```

פונקציות:

```
function XxorK=Xxork(num_of_enc,bytes_of_AES,AES_key_bit,X)
Xxork=zeros(num_of_enc,AES_key_bit,bytes_of_AES);
 for k=1:bytes_of_AES
         for i=1:num_of_enc
              for j=1:AES_key_bit
               % bitxor-ing the all key column of X (all key Possibility of aes)
                 Xxork(i,j,k) = bitxor(X(i,k),j-1);

              end
         end
 end
XxorK=Xxork;
end
function B=b(num_of_enc,bytes_of_AES,AES_key_bit,XxorK)
b=zeros(num_of_enc,AES_key_bit,bytes_of_AES);
 for k=1:bytes_of_AES
         for i=1:num_of_enc
              for j=1:AES_key_bit
                 % pass the XxorK to S-BOX transformation
                 b(i,j,k) = SBOX_table(XxorK(i,j,k)+1);

              end
         end
 end
B=b;
End
function HW_B=HW(B,bytes_of_AES,AES_key_bit,num_of_enc)
hw = zeros(num_of_enc,AES_key_bit,bytes_of_AES);
 for k=1:bytes_of_AES
        for i = 1:AES_key_bit

           hw(:,i,k) = sum(dec2bin(B(:,i,k)).' == '1' );
        end
 end
 HW_B=hw;
end
function HD_B=HD(B,bytes_of_AES,AES_key_bit,num_of_enc,XxorK)
hd = zeros(num_of_enc,AES_key_bit,bytes_of_AES);
 for k=1:bytes_of_AES
       for i = 1:AES_key_bit

            hd(:,i,k) = sum(dec2bin(bitxor((B(:,i,k)),(XxorK(:,i,k)).' == '1' )));
       end
 HD_B=hd;
 end
end

function conv = SBOX_table (index)
    % S-box
    SBOX=[099 124 119 123 242 107 111 197 048 001 103 043 254 215 171 118 ...
          202 130 201 125 250 089 071 240 173 212 162 175 156 164 114 192 ...
          183 253 147 038 054 063 247 204 052 165 229 241 113 216 049 021 ...
          004 199 035 195 024 150 005 154 007 018 128 226 235 039 178 117 ...
          009 131 044 026 027 110 090 160 082 059 214 179 041 227 047 132 ...
          083 209 000 237 032 252 177 091 106 203 190 057 074 076 088 207 ...
          208 239 170 251 067 077 051 133 069 249 002 127 080 060 159 168 ...
          081 163 064 143 146 157 056 245 188 182 218 033 016 255 243 210 ...
          205 012 019 236 095 151 068 023 196 167 126 061 100 093 025 115 ...
          096 129 079 220 034 042 144 136 070 238 184 020 222 094 011 219 ...
          224 050 058 010 073 006 036 092 194 211 172 098 145 149 228 121 ...
          231 200 055 109 141 213 078 169 108 086 244 234 101 122 174 008 ...
          186 120 037 046 028 166 180 198 232 221 116 031 075 189 139 138 ...
          112 062 181 102 072 003 246 014 097 053 087 185 134 193 029 158 ...
          225 248 152 017 105 217 142 148 155 030 135 233 206 085 040 223 ...
          140 161 137 013 191 230 066 104 065 153 045 015 176 084 187 022];
    conv = SBOX(index);
end
function trueCR =CR(input_matrix,trueKey)
temp = abs(input_matrix);
max_corr = max(temp);
firsMAXtCorr=max(max_corr); %find the first max correlation
```

```
firstmaxPlace=find(max_corr==firsMAXtCorr); %first max correlation place
max_corr(firstmaxPlace)=0; %Delete the first maximum
secondMAXcCorr=max(max_corr); %find the second max correlation
max_corr(firstmaxPlace)=firsMAXtCorr; %return first max correlation to the
max_corr
cr=firsMAXtCorr/secondMAXcCorr;
trueCR=max_corr(trueKey)/firsMAXtCorr;
if (trueCR == 1)
    trueCR = cr;
end
%correlation ratio>1 attack was successfull(Continues to be higher than 1)

end
```

פוסטר הפרויקט:

הנדסת חשמל

עוז יוסף , עומר אביב
מנחה: ד"ר יואב ויצמן

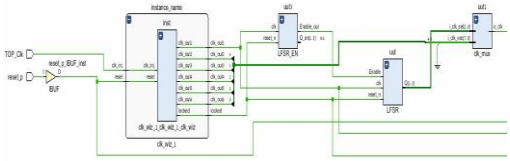
הטמעה של הגנות חומרה מבוססות רנדומיזציה בזמן במערכת קריפטוגרפית לצורך שיבוש זליגת המידע ומניעה התקפות ערוץ צד.

מהות הפרויקט הוא להגן ולאבטח על מערכות חומרה אל מול התקפות ערוץ צד המבוססות על צריכת ההספק של המערכת או התקפות אלקטרומגנטיות המאיימות על חילוץ המידע הסודי. בעזרת מימוש מערכת הגנה של פיזור האינפורמציה בזמן ע"י שינויים במימוש החומרה של מערכת ההצפנה על רכיב FPGA ,נקבל מערכת המוגנת התקפות מטרתן חילוץ המידע הסודי.

אלגוריתם ההצפנה Advanced Encryption Standard (AES) נמצא חסין באופן מעשי מפני Classical Cryptanalysis הכולל התקפת כוח גס (Brute Force) שזה למעשה ניסיון חילוץ של המפתח הסודי באמצעות מעבר סידרתי על כל הצירופים השונים ו Mathematical Attack הכולל יישום אלגוריתמים מתמטיים שונים, אך עדיין פגיע בפני התקפות ערוץ צד המנצלת את פליטת ההספק אשר נפלט מהמערכת בזמן העבודה של פעולות הליבה קריפטוגרפיות בצורה זו ניתן להשיג את המידע הסודי המוצפן. לכן בפרויקט שלנו ניצור אמצעי הגנה ונממש מערכת ייחודית בשם RandomClock המשפרת את יעילות מערכת הAES וגורמת לפיזור האינפורמציה בזמן ובכך מונע מהתוקף להשיג את המידע הסודי .

איור 1

באיור 1 נראה את כלל סבבי מערכת ההצפנה AES לבדה , הצפנה הכוללת 14 סבבי הצפנה שונים שבכל אחד מתבצעות הפעולות המתמטיות אשר מצפינות את המידע הסודי .
באיור 2 נראה את מערכת ה RandomClock הכוללת רכיב PLL המספק במוצאו שעונים בעלי פאזות שונות ומבוררות בעזרת MUX גלובלי ורגיסטר LFSR אל תור מערכת הAES .

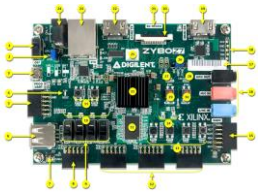


איור 2

מטרות הפרויקט :

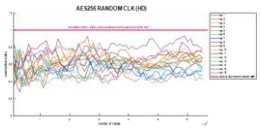
1.מימוש של מערכת הצפנה על רכיב FPGA שהיא מסוג הצפנה סימטרית בסיסית AES בדיקה ואימות של המימוש שבוצע. תכנון הגנות בעת המימוש שמונעות אפשרות לביצוע התקפות ערוץ צד .
2.מדידות של עכבות הפליטה האלקטרומגנטית תוך כדי הרצה של מילות הצפנה על המיקרו מעבד ושמירה של האותות הנמדדים באמצעות החיישן בזיכרון האוסילוסקופ .
3.אנליזות של האותות הנמדדים – כתיבה של קוד Python/MATLAB המסוגל לטעון את האותות הנמדדים ולבצע עיבוד לאותות שנמדדו . חישוב הקורלציה בין האותות הנמדדים ומציאת המפתח בעל הקורלציה הגבוהה ביותר .
4.מימוש אלגוריתם פיזור האינפורמציה בזמן ומדידה חוזרת של האותות ובדיקה של יכולות התקיפה לאחר המימוש.

את הפרויקט מימשנו על גבי כרטיס FPGA מסוג Xilinx Digilent Zybo Zynq-7000 (איור 3) אשר בעזרתו נבדוק את מימושי החומרה להפקת תהליכי ההצפנה אשר תכננו.
בכדי לבדוק את יעילות ההגנה המשופרת אשר מימשנו בפרויקט , ביצענו התקפת ערוץ צד יזומה על הכרטיס בה מדדנו את פליטת ההספק מהכרטיס בזמן ההצפנה ומדדנו את הקורלציה בין צריכת ההספק לניחושי המפתח הסודי .

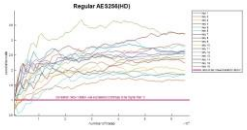


איור 3

במידה והצלחנו להגן על הרכיב , ניחוש המפתח הנכון לא יהיה בעל הקורלציה הגבוהה ביותר . את איכות הגילויים של האלגוריתם ניתן להעריך לפי מדד ה -CR (Correlation Ratio) או טיב הקורלציה , ככל ש CR>1 ונשאר מעל ערך זה , כך לניחוש יש סיכוי גדול יותר להיות הניחוש הנכון למפתח הסודי .



גרף 1



גרף 2

מגרף 1 ניתן לראות את המערכת המוגנת בעזרת טכנולוגיית ה RandomClockAES שבה מדד הCR נותר נמוך מ-1 והמערכת נשאר מוגנת ולא נפרצה.
בגרף 2 ניתן לראות את מערכת הAES לבדה שבה נראה בבירור שהתגלו כל המפתחות הסודיים והמערכת נפרצה.
כעת נציג טבלה המראה את שיפור יעילות וביצועי מערכת ההגנה .

Type of Implementation	Regular AES	Regular AES	RandomClock AES	RandomClock AES	Improvement	Improvement
Type of attack	HW	HD	HW	HD	HW	HD
Measurement to Disclosure	≈4001	≈301	>65,280	>65,280	>1,531%	>21,584%

טבלה 1 – ריכוז שיפור הביצועים