Intro. to Cryptography
Programming Assignment - Salsa*

TA Yitang
Email Address: yitang@cs.nctu.edu.tw
TA Office Hour: Wednesday 12:30 - 14:00 at EC119
            Friday 12:30 - 14:00 at EC119
Salsa20

- A software stream cipher standard of eSTREAM.
- **Key**: 32 bytes (4 words) or 16 bytes (2 words)
- **Nonce**: 8 bytes (2 words)
- **Sequence number**: 8 bytes (2 words)
- **Constants**: 16 bytes (4 words)
- Generating $2^{70}$ bytes sequence. (64 bytes a time)
Salsa20- Matrix

- For the 8-word-key, build a matrix as follow:

  \[
  \begin{array}{cccc}
  \text{const}[0] & \text{key}[0] & \text{key}[1] & \text{key}[2] \\
  \text{key}[3] & \text{const}[1] & \text{nonce}[0] & \text{nonce}[1] \\
  \text{s\_num}[0] & \text{s\_num}[1] & \text{const}[2] & \text{key}[4] \\
  \text{key}[5] & \text{key}[6] & \text{key}[7] & \text{const}[3]
  \end{array}
  \]
Salsa20- Operations

- **Sum (+):** The sum of two words $a$ and $b$ is $a + b \mod 2^{32}$.
- **XOR ($\oplus$):** A bitwise operation between two words.
- **Left rotation ($\lll$):** A bitwise operation of a word.
  - example 1: $0x80208010 \lll 1 = 0x00410021$
  - example 2: $0x80208010 \lll 32 = 0x80208010$
Salsa20- Quarterround

- $y = (y_0, y_1, y_2, y_3)$, $quarterRound(y) = (z_0, z_1, z_2, z_3)$
  - $z_1 = y_1((y_0 + y_3) \ll 7)$
  - $z_2 = y_2((z_1 + y_0) \ll 9)$
  - $z_3 = y_3((z_2 + z_1) \ll 13)$
  - $z_0 = y_0((z_3 + z_2) \ll 18)$
Salsa20- Rowround

Definition

If \( y = (y_0, y_1, y_2, y_3, \ldots, y_{15}) \) then \( \text{rowround}(y) = (z_0, z_1, z_2, z_3, \ldots, z_{15}) \) where

\[
(z_0, z_1, z_2, z_3) = \text{quarterround}(y_0, y_1, y_2, y_3),
(z_5, z_6, z_7, z_4) = \text{quarterround}(y_5, y_6, y_7, y_4),
(z_{10}, z_{11}, z_8, z_9) = \text{quarterround}(y_{10}, y_{11}, y_8, y_9),
(z_{15}, z_{12}, z_{13}, z_{14}) = \text{quarterround}(y_{15}, y_{12}, y_{13}, y_{14}).
\]
Definition

If $x = (x_0, x_1, x_2, x_3, \ldots, x_{15})$ then columnround$(x) = (y_0, y_1, y_2, y_3, \ldots, y_{15})$

where

$(y_0, y_4, y_8, y_{12}) = \text{quarterround}(x_0, x_4, x_8, x_{12}),$
$(y_5, y_9, y_{13}, y_1) = \text{quarterround}(x_5, x_9, x_{13}, x_1),$
$(y_{10}, y_{14}, y_2, y_6) = \text{quarterround}(x_{10}, x_{14}, x_2, x_6),$
$(y_{15}, y_3, y_7, y_{11}) = \text{quarterround}(x_{15}, x_3, x_7, x_{11}).$
Salsa20- Doubleround

columnRound(rowRound( ))

const[0]  key[0]  key[1]  key[2]
Salsa20

\[
S = \text{doubleround}^{10}(\text{const[0]} \quad \text{key[0]} \quad \text{key[1]} \quad \text{key[2]} \\
\text{key[3]} \quad \text{const[1]} \quad \text{nonce[0]} \quad \text{nonce[1]} \\
\text{s_num[0]} \quad \text{s_num[1]} \quad \text{const[2]} \quad \text{key[4]} \\
\text{key[5]} \quad \text{key[6]} \quad \text{key[7]} \quad \text{const[3]})
\]

\[
+ \quad \text{const[0]} \quad \text{key[0]} \quad \text{key[1]} \quad \text{key[2]} \\
\text{key[3]} \quad \text{const[1]} \quad \text{nonce[0]} \quad \text{nonce[1]} \\
\text{s_num[0]} \quad \text{s_num[1]} \quad \text{const[2]} \quad \text{key[4]} \\
\text{key[5]} \quad \text{key[6]} \quad \text{key[7]} \quad \text{const[3]}
\]

To generate enough bytes, just increase the sequence number and do the above again.
Assignment - Modified Salsa

- **Key**: 32 bytes (8 words)
- **Nonce**: 16 bytes (4 words)
- **Sequence number**: 4 bytes (1 word)
- **Constants**: 32 bytes (8 words)
Modified Salsa- Matix

Modified Salsa- Operations

- **Sum (+):** The sum of two words \( a \) and \( b \) is \( a + b \mod 2^{32} \).
- **XOR (⊕):** A bitwise operation between two words.
- **Left rotation (<<<):** A bitwise operation of a word.
- **NOT (~):** A bitwise operation of a word.
  - example 1: \(~0x00000001=0xFFFFFFF0\)
  - example 1: \(~0x123456AB=0xEDCBA954\)
Modified Salsa- Round

\[
\begin{align*}
\text{round}(y_0,y_1,y_2,y_3,y_4\ldots,y_7) &= (z_0,z_1,z_2,z_3,\ldots,z_7) \\
z_3 &= (~y_3 \oplus y_6) + ((y_4 \oplus y_5 \oplus ~y_1)<<<7) \\
z_4 &= (~y_4 \oplus y_7) + ((y_5 \oplus y_6 \oplus ~y_2)<<<11) \\
z_5 &= (~y_5 \oplus y_0) + ((y_6 \oplus y_7 \oplus ~z_3)<<<13) \\
z_6 &= (~y_6 \oplus y_1) + ((y_7 \oplus y_0 \oplus ~z_4)<<<17) \\
z_7 &= (~y_7 \oplus y_2) + ((y_0 \oplus y_1 \oplus ~z_5)<<<19) \\
z_0 &= (~y_0 \oplus z_3) + ((y_1 \oplus y_2 \oplus ~z_6)<<<23) \\
z_1 &= (~y_1 \oplus z_4) + ((y_2 \oplus z_3 \oplus ~z_7)<<<29) \\
z_2 &= (~y_2 \oplus z_5) + ((z_3 \oplus z_4 \oplus ~z_0)<<<31)
\end{align*}
\]
If $y = (y_0, y_1, \ldots, y_{63})$ then bigRound($y$) = $(z_0, z_2, \ldots, z_{63})$ where

$(z_6, z_{52}, z_{40}, z_{26}, z_1, z_{51}, z_{47}, z_{29}) = \text{round}(y_6, y_{52}, y_{40}, y_{26}, y_1, y_{51}, y_{47}, y_{29})$

$(z_{11}, z_{57}, z_{37}, z_{23}, z_{12}, z_{62}, z_{34}, z_{16}) = \text{round}(y_{11}, y_{57}, y_{37}, y_{23}, y_{12}, y_{62}, y_{34}, y_{16})$

$(z_{50}, z_0, z_{28}, z_{46}, z_{53}, z_7, z_{27}, z_{41}) = \text{round}(y_{50}, y_0, y_{28}, y_{46}, y_{53}, y_7, y_{27}, y_{41})$

$(z_{63}, z_{13}, z_{17}, z_{35}, z_{56}, z_{10}, z_{22}, z_{36}) = \text{round}(y_{63}, y_{13}, y_{17}, y_{35}, y_{56}, y_{10}, y_{22}, y_{36})$

$(z_{24}, z_{42}, z_{54}, z_4, z_{31}, z_{45}, z_{49}, z_3) = \text{round}(y_{24}, y_{42}, y_{54}, y_4, y_{31}, y_{45}, y_{49}, y_3)$

$(z_{21}, z_{39}, z_{59}, z_9, z_{18}, z_{32}, z_{60}, z_{14}) = \text{round}(y_{21}, y_{39}, y_{59}, y_9, y_{18}, y_{32}, y_{60}, y_{14})$

$(z_{44}, z_{30}, z_2, z_{48}, z_{43}, z_{25}, z_5, z_{55}) = \text{round}(y_{44}, y_{30}, y_2, y_{48}, y_{43}, y_{25}, y_5, y_{55})$

$(z_{33}, z_{19}, z_{15}, z_{61}, z_{38}, z_{20}, z_8, z_{58}) = \text{round}(y_{33}, y_{19}, y_{15}, y_{61}, y_{38}, y_{20}, y_8, y_{58})$
Modified Salsa

\[ S = \text{bigRound}^{16}(\ldots) \]

To generate enough bytes, just increase the sequence number by 1 and do the above again.
Program

Language:
- C/C++ are recommended.
- If you want to use other language, please let TA know first and make sure CS Workstation can run/compile it.

Testing platform:
- CS Workstation (linux3/bsd3)

Input:
- One line standard input as below:
  
  key_file nonce_file file_in file_out

  - key_file: The name of the file in which the key is stored.
  - nonce_file: The name of the file in which the initial vector is stored.
  - file_in: The input file name.
  - file_out: The output file name.

Output:
- No standard output
- Using Modified Salsa to encrypt(or decrypt) the file <file_in>. And the result should be stored as a file named <file_out>.
Handin

- Upload to e3:
  - Source code named in the following format
    `<student id>.<ext>`  ex:0123123.cpp
  - Makefile (if necessary)

- You don’t need to compress them.
Deadline

- 23:59:59 on 2015/01/07 (Wed.)
- No late submission.
- Do NOT copy others. Plagiarism gives nothing else to you but a ‘0’ as your final score.
Scoring

- Submission on time:
  - Passing basic testing cases: 60 points (see attachment file)
  - Passing secret testing cases: 40 points (revealed after scoring)
  - Other situations: According to the situation, you will lose some points.
- Late submission: 0 point
- Plagiarism: 0 point in this course
Q1 - How to XOR Plaintext?

- In little endian:
  - If the word is 0x12345678, the first byte should be 0x78.
  - Most machine is using this encoding method, you don’t need to change anything but take the word as 4 bytes.
  - Ex: You can compile the following C code to check.
    unsigned int a=0x12345678;
    unsigned char *p=(char*)&a;
    printf("a[0]=%02X, a[1]=%02X, a[2]=%02X, a[3]=%02X",a[0], a[1], a[2], a[3]);

- So, if a four-byte-plaintext (0x12, 0x34, 0x56, 0x78) XOR the word S=0x12345678. The ciphertext should be (0x12 XOR 0x78, 0x34 XOR 0x56, 0x56 XOR 0x34, 0x78 XOR 0x12) = (0x6A, 0x62, 0x62, 0x6A)
Q2- The Sequence Number?

- You can take the Salsa\* as a hash function.
  - The input is the $8 \times 8$ words matrix.
  - The output $8 \times 8$ matrix can be take as 256 bytes array.
  - If the file size is greater than 256 byte, for the input of the next run, you have to make a new $8 \times 8$ matrix in which the sequence number is added by 1.
  - The sequence number should be initialized to 0 in the beginning.
Q3- Key File

- The key size is 32 bytes (8 words).
- The key file stores the key `key[8]` as following:

  ```
  key[0]
  key[1]
  key[2]
  key[3]
  key[4]
  key[5]
  key[6]
  key[7]
  ```

  One word in hexadecimal each line (in big endian, meaning you can easily read to your program).

- And so is the nonce file (4 words).
Contact TA

- Email by e3.
- Email to yitang@cs.nctu.edu.tw
- Office hour: Wednesday 12:30 - 14:00 at EC119
  Friday 12:30 - 14:00 at EC119