Problem Set 7
Course **Secure Channels**
(Summer Term 2018)

Bauhaus-Universität Weimar, Chair of Media Security
Prof. Dr. Stefan Lucks, Eik List
URL: `http://www.uni-weimar.de/de/medien/professuren/mediensicherheit/teaching/`

Due Date: 12 July 2018, 11:00 AM, via email to `eik.list(at)uni-weimar.de`.

Choose *one* part, theoretical *or* practical.

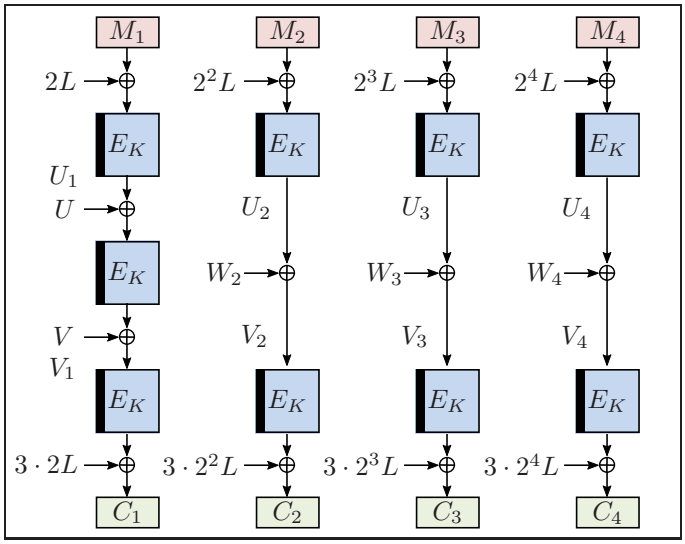
Theoretical Part

Task 1 – Wide-block Encryption (8 Credits)

Below you can find a schematic illustration of a variant of the Encrypt-Mix-Encrypt [1] wide-block cipher. It takes a message $M \in (\{0,1\}^n)^*$ and computes a ciphertext of equal length as the message. Note that we define it for messages whose length is any multiple of n bit. Otherwise, the encryption returns $\perp$.

E_K is a secure block cipher, and $K, L \twoheadleftarrow \mathcal{K}$ independent random keys. In the following, all multiplications are in $\mathbb{GF}(2^n)$. We define the encryption $\mathcal{E}_{K,L}(M)$ of a message $M = (M_1, \ldots, M_m)$ as given below. The decryption works analogously.

1: **function** $\mathcal{E}_{K,L}(M)$
2: **if** $|M| \bmod n \neq 0$ **then**
3: **return** $\perp$
4: $(M_1, \ldots, M_m) \xleftarrow{n} M$
5: **for** $i = 1$ to m **do**
6: $U_i \leftarrow E_K(M \oplus 2^{i-1}L)$
7: $U \leftarrow \bigoplus_{i=2}^{m} U_i$
8: $V_1 \leftarrow E_K(U_1 \oplus U)$
9: $W \leftarrow U_1 \oplus U \oplus V_1$
10: **for** $i = 2$ to m **do**
11: $W_i \leftarrow W$
12: $V_i \leftarrow U_i \oplus W_i$
13: $V \leftarrow \bigoplus_{i=2}^{m} V_i$
14: $V_1 \leftarrow V_1 \oplus V$
15: **for** $i = 1$ to m **do**
16: $C_i \leftarrow E_K(V_i) \oplus 3 \cdot 2^{i-1}L$
17: **return** $(C_1 \parallel \ldots \parallel C_m)$



a) *Either* describe briefly an attack on its RoR-CCA security *or* explain briefly why it is secure.

b) Try the following: In $\mathbb{GF}(2^8)$ with irreducible modulus polynomial $\mathtt{x}^8 + \mathtt{x}^4 + \mathtt{x}^3 + \mathtt{x}^1 + \mathtt{1}$, calculate $S(L) = 2^8L \oplus 2^4L \oplus 2^3L \oplus 2L \oplus L$ for some values of L of your choice. Briefly describe your observations.

c) The original EME uses $W_i = 2^i W$ for $i = 2, 3, \ldots$ in the XORs in the middle layer. Though, it was defined only for at most $m = n$ blocks. For our variant that is defined for messages of arbitrary number of blocks m, *either* briefly an attack on its RoR-CCA security *or* explain briefly why it is secure. (*Hint:* Task b) could be there for a purpose.)

Task 2 – Encode-then-Encipher (4 Credits)

Consider the original EME, i.e., the construction above with $W_i = 2^i \cdot W$ and to $m < n$ blocks. We turn it into a deterministic authenticated encryption scheme as follows: For every message $M = (M_1, \ldots, M_m)$, we append a final block of n zero bits before encryption. This block is verified at decryption and is not returned.

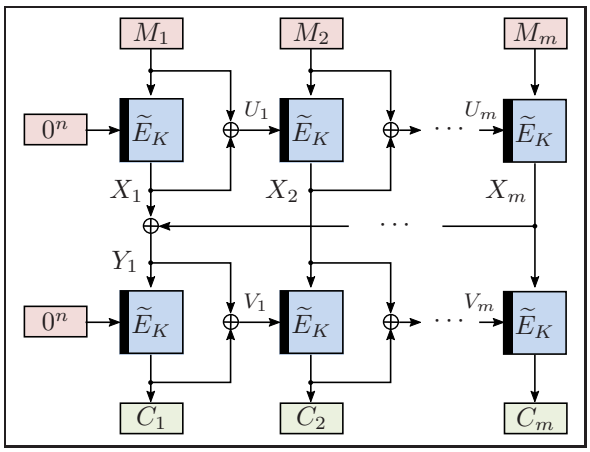
Either describe briefly an attack on its RoR-CCA security *or* explain briefly why it is secure.

function $\mathcal{AE}_{K,L}(M)$
  if $|M| \bmod n \neq 0$ **then**
    return $\perp$
  if $|M| \geq (n-1) \cdot n$ **then**
    return $\perp$
  $M' \leftarrow M \,\|\, 0^n$
  return $\mathcal{E}_{K,L}(M')$

function $\mathcal{AD}_{K,L}(C')$
  if $|C'| \bmod n \neq 0$ or $|C'| \geq n^2$ **then**
    return $\perp$
  $M' \leftarrow \mathcal{D}_{K,L}(C')$
  $(M_1, \ldots, M_{m+1}) \xleftarrow{n} M'$
  if $M_{m+1} \neq 0^n$ **then**
    return $\perp$
  return $M_1 \,\|\, \ldots \,\|\, M_m$

Task 3 – Turning On-line Ciphers Off (4 Credits)

Below, you can find an off-line cipher from two stacked executions of McOE. Let $\widetilde{E}_K$ be a secure tweakable block cipher that takes n-bit tweaks. The construction below is defined for messages whose length is an arbitrary multiple of n bits: $M \in (\{0,1\}^n)^*$. The XOR in the middle is omitted if $|M| = n$. *Either* describe briefly an attack on its RoR-CCA security *or* explain briefly why it is secure.



Practial Part

For all tasks below: This is real-life work from real-life people. Third-party authors may be very good at what they do and we all do mistakes. All tasks are not intended as advertisement or critique of any kind. Let's keep humble as always.

Task 4 – Auditing Homebrew Crypto (6 Credits)
Designing secure crypto is hard. Then, read autonomously into the MTProto 2.0 protocol: `https://core.telegram.org/mtproto`. *Either* describe briefly attack or attack vectors *or* explain why it provides Level-4 stateful-AE security.

Note: There exist earlier analyses [6, 7] on MTProto 1.0; you can research and reference them if they still apply.

Task 5 – Auditing Homebrew-crypto Implementations (6 Credits)
Implementing secure crypto is even harder. You need to account for a plethora of potential issues, such as correctness of the algorithm, input sanitizing, memory security, correct use of nonces, secure random-number generation, avoiding algorithmic pitfalls, absence of side channels (timing, failure reason) etc.

Get an impression into a proper design and categorization of security audits, e.g., [2, Section 3]. Read autonomously into [4] (you can find many more sources from the wiki that are beyond this task). Then, apply your knowledge by auditing the Python port of the Telegram protocol: `pyrogram`. Take a look into the cryptographic implementation of its `crypto` subpackage: [5]. Perform an audit of the `crypto` subpackage from the state at 2018-06-29.

(Note: There may exist tools to automate the task [3], they are not necessary here.)

Task 6 – Secure Protocol (6 Credits)
Design yourselves a messaging protocol that provides Level-4 stateful AE security for the same purpose (secure instant messaging). You can assume that session keys have been negotiated before securely. Simply reusing an existing higher-level protocol (like TLS etc.) is *NOT* an allowed option for this task.

- Choose an appropriate AE scheme and primitives of your choice. Provide a brief rationale why you chose which component.
- Explain which low-level protocol(s) (TCP/UDP/WebRTC etc.) you would use for transport and why.

References

[1] Shai Halevi, Phillip Rogaway: A Parallelizable Enciphering Mode. CT-RSA 2004: 292-304.

[2] Alex Balducci, Sean Devlin, and Tom Ritter. Cryptographic Review Cryptography Services Final Report — Open Crypto Audit Project TrueCrypt. 2015. `https://opencryptoaudit.org/reports/TrueCrypt_Phase_II_NCC_OCAP_final.pdf`.

[3] Bandit Developers on github `https://github.com/PyCQA/bandit`.

[4] Coding rules: https://cryptocoding.net/index.php/Coding_rules, 2016.

[5] delivrance. Pyrogram. Telegram MTProto API Client Library for Python. Crypto Subpackage. https://github.com/pyrogram/pyrogram/tree/develop/pyrogram/crypto

[6] Jakob Jakobsen, Claudio Orlandi: On the CCA (in)Security of MTProto. SPSM@CCS 2016: 113-116. Full version at https://eprint.iacr.org/2015/1177.pdf.

[7] Nadim Kobeissi, Karthikeyan Bhargavan, Bruno Blanchet: Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach. EuroS&P 2017: 435-450 http://prosecco.gforge.inria.fr/personal/bblanche/publications/KobeissiBhargavanBlanchetEuroSP17.pdf