

Homeworks

Computer and Network Security

Emilio Coppa

Basic rules

- 7 homeworks released during the course
- submit solution for at least 4 homeworks to get a bonus of 2 points in the final exam
- the bonus can be used only if your final grade is equal or higher than 24
- program valid until Sept 2021 (exam sessions: Jan, Feb, Jun, July, Sept);
undergraduate/Erasmus/other master's students can participate
- Compilatio plagiarism detection, checking the similarity with web documents and with other
consigned homeworks
- don't post or share to anybody the assigned homeworks

Homework submission

1. Report (PDF):

- Discuss in detail the solution to the homework (max 15 pages, no need to be verbose if not useful)
- PDF generated using LaTeX
- title (homework name/ID), subtitle (CNS Course Sapienza), Author (first name, last name, student ID), date (deadline), no abstract
- use article class, 11 pt
- language: English
- if pictures needed use `\includegraphics`

2. ZIP archive: a compressed .zip archive containing

- source code of the solution
- auxiliary scripts
- source code of the report (latex files, bib files, figures, etc.)

Homework submission (2)

Submission form:

<https://forms.gle/pnUWoWunRe1YxRaH9>

(to fill the form use the institutional mail from Sapienza or, if you do not have one yet, a personal mail from Google)

Homeworks will be evaluated in a binary form: **accept/reject**
A reject is notified via mail and can happen at any time during the year (even during the exam where you plan to use the bonus).

Homeworks: deadlines (subject to changes!)

| ID | Released | Deadline |
|----|------------|------------------------------------|
| 1 | 06/10/2020 | Dynamic deadline: up to 31/12/2020 |
| 2 | 20/10/2020 | 30/10/2020 |
| 3 | 30/10/2020 | 20/11/2020 |
| 4 | 30/10/2020 | 20/11/2020 |
| 5 | 17/11/2020 | 30/11/2020 |
| 6 | 27/11/2020 | 10/12/2020 |
| 7 | 08/12/2020 | 20/12/2020 |

Homework #1: solve two crypto challenges in a CTF

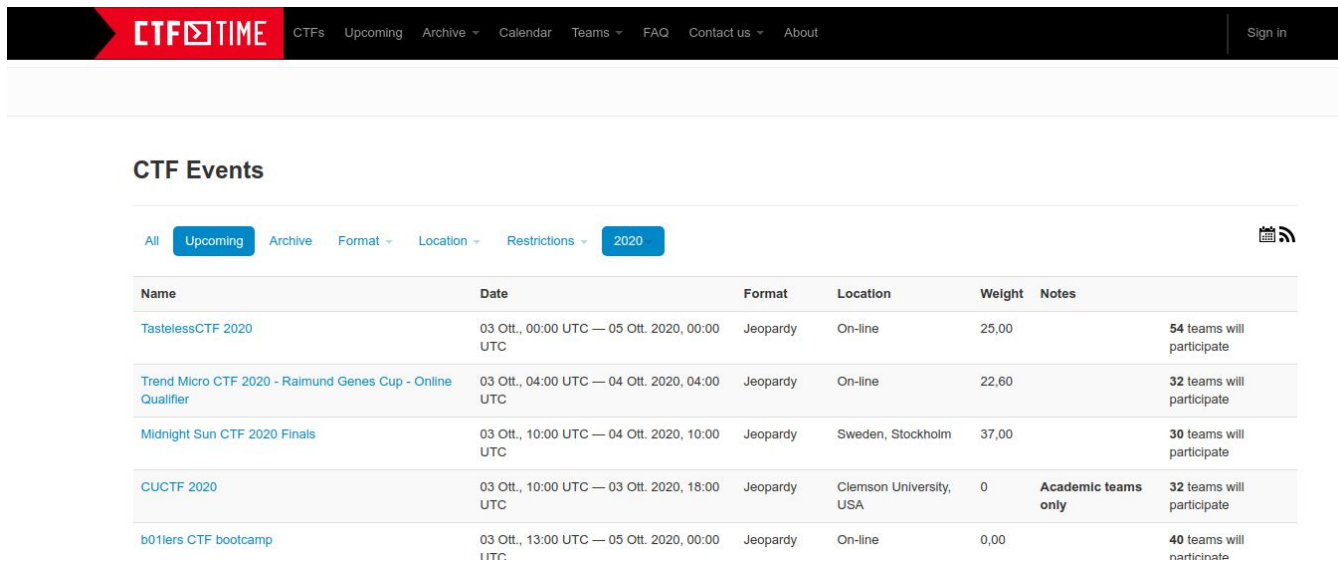
In computer security **Capture the Flag (CTF)**, "flags" are secrets hidden in purposefully-vulnerable programs or websites. Competitors steal flags either from other competitors (attack/defense-style CTFs) or from the organizers (jeopardy-style challenges). Several variations exist, including hiding flags in hardware devices.

Security CTFs are usually designed to serve as an educational exercise to give participants experience in securing a machine, as well as conducting and reacting to the sort of attacks found in the real world (i.e., bug bounty programs in professional settings). Classic activities include reverse-engineering, network sniffing, protocol analysis, system administration, programming, **cryptoanalysis**, writing exploits

Homework #1: solve two crypto challenges in a CTF (2)

Rules:

- Choose one CTF **jeopardy** competition listed on [CTFtime.org](https://ctftime.org)



The screenshot shows the CTFtime.org website. The header is black with a red 'CTF TIME' logo on the left and navigation links (CTFs, Upcoming, Archive, Calendar, Teams, FAQ, Contact us, About) and a 'Sign in' button on the right. Below the header is a 'CTF Events' section with filters: All, Upcoming (selected), Archive, Format, Location, Restrictions, and 2020 (selected). A table of events is displayed below the filters.

| Name | Date | Format | Location | Weight | Notes |
|---|--|----------|-------------------------|--------|--|
| TastelessCTF 2020 | 03 Oct., 00:00 UTC — 05 Oct. 2020, 00:00 UTC | Jeopardy | On-line | 25,00 | 54 teams will participate |
| Trend Micro CTF 2020 - Raimund Genes Cup - Online Qualifier | 03 Oct., 04:00 UTC — 04 Oct. 2020, 04:00 UTC | Jeopardy | On-line | 22,60 | 32 teams will participate |
| Midnight Sun CTF 2020 Finals | 03 Oct., 10:00 UTC — 04 Oct. 2020, 10:00 UTC | Jeopardy | Sweden, Stockholm | 37,00 | 30 teams will participate |
| CUCTF 2020 | 03 Oct., 10:00 UTC — 03 Oct. 2020, 18:00 UTC | Jeopardy | Clemson University, USA | 0 | Academic teams only 32 teams will participate |
| b01lers CTF bootcamp | 03 Oct., 13:00 UTC — 05 Oct. 2020, 00:00 UTC | Jeopardy | On-line | 0,00 | 40 teams will participate |

Entry level CTF are OK.

Homework #1: solve two crypto challenges in a CTF (3)

- Register on CTftime.org, register a team
- Register on the CTF website:
 - you should play alone using the team created on CTFTIME.org
 - you should have confirmation email of the registration, showing your nickname and the event
- **Play the CTF:** solve at least two challenges that are “tagged” by the organizer as “crypto” or any other topic covered by the course (see syllabus)
- Write a **write-up** for each challenge and post it on CTftime.org (see [this FAQ](#))

Homework #1: solve two crypto challenges in a CTF (4)

- Write a PDF report using LaTeX, integrating the two write-ups, screenshot of the confirmation mail from the CTF event, and screenshot about correctness on the flag (points on the CTF website). At the beginning of the report, provide details about the CTF event: name, URL, URL on CTFTIME.org, start date, end date.
- Submit as a homework #1:
 - the PDF report
 - a ZIP archive containing any source code/program/script related to the challenges
- Deadline: 48 hours after the end of the CTF competition

Homework #1: solve two crypto challenges in a CTF (5)

You think it will like this:



Homework #1: solve two crypto challenges in a CTF (6)

It will be like this:



Homework #1: solve two crypto challenges in a CTF (7)

The goal of this homework is to have fun while learning crypto!

Playing CTF is the best way of learning cybersecurity!

Homework #2: AES implementation

Goals of the homework:

1. implementation of AES-128
2. extend implementation adding operation modes ECB, CBC, CFB, OFB, CTR
3. experimental comparison of your implementation wrt performance to a well-known implementation

The report should discuss how you have structured the implementation, motivate any design choice, and discuss the performance comparison.

Homework #2: AES implementation (2)

Implementation of AES-128:

- you can use any programming language (Python is suggested)
- different approaches (slower to faster, interesting to boring):
 - perform computations in $GF(256)$ for `SubBytes()` and `MixColumns()`: you learn a lot but it will take a bit of time, you should use some library (e.g., [SageMath](#)/Python) to deal with GF.
 - use precomputed S-Box for `SubBytes()`, `xtime()` for `MixColumns()`
 - use precomputed S-Box for `SubBytes()`, precomputed lookup table for `MixColumns()`
 - single lookup table for all steps except `roundKey()`

Homework #2: AES implementation (3)

Implementation of AES-128:

- **useful documents (Resource page in Piazza):**
 - FIPS-197: Advanced Encryption Standard (AES): see examples in Appendix C
 - Example of MixColumn step
 - The Design of Rijndael by Joan Daemen and Vincent Rijmen: see section 4.1
- **many public implementations of AES: it is ok to “look” at them to learn some details... but you should not do cut-and-paste code...**
- **you need to implement both encryption and decryption**

Homework #2: AES implementation (4)

Extend implementation adding operation modes ECB, CBC, CFB, OFB, CTR:

- padding is needed: one easy approach is [PKCS#7 padding](#), but also other approaches are fine (your choice).
- see document “Recommendation for Block Cipher Modes of Operation by NIST” on the Resource page in Piazza:
 - CFB can done in different ways (parameter s , choose one value)
 - check out examples in Appendix F

Homework #2: AES implementation (5)

Experimental comparison of your implementation wrt performance to a well-known implementation:

- choose one “standard” and “well-known” implementation of AES that provides at least three operation modes (3 out of 5: ECB, CBC, CFB, OFB, CTR). For instance, you can easily use [PyCryptoDome](#) (even if your implementation is not in Python!). Most languages have a library providing AES. [OpenSSL](#) (or LibreSSL) is another good option.
- try to validate the correctness of your implementation by comparing ciphertexts generated by the two implementations. **Notice that the well-known implementation may use a different padding scheme and different “parameters”, hence it is fine if you do not get the same output for all modes but you should (try) to investigate the reason.**

Homework #2: AES implementation (6)

Experimental comparison of your implementation wrt performance to a well-known implementation:

- perform an experimental comparison between your implementation and the the “well-known” implementation:
 - consider three different file sizes: 1KB, 100KB, 10MB
 - compare performance of different modes in encryption and decryption: a standard metric reported for ciphers is the throughput
 - use different charts/plots in the report to show the comparison
- It is OK if your implementation is way slower than the “well-known” implementation. **The ultimate goal of this homework is to get an idea of how AES works, how different modes work, and have a bit of familiarity with one real-world AES implementation.**

Homework #3: playing with PK schemes

Goals of the homework:

1. develop your own implementation of RSA (encryption/decryption)
2. experimental comparison of your implementation wrt performance to a real-world implementation of RSA
3. experimental comparison of your implementation wrt performance to a real-world implementation of AES

The report should discuss how you have structured the implementation, motivate any design choice, usage examples, and discuss the performance comparisons.

Homework #3: playing with PK schemes (2)

Implementation of PK schemes:

- you can use any programming language to implement RSA (Python is suggested)

Homework #3: playing with PK schemes (3)

- Some aspects that you have to figure out by yourself:
 - how to find & choose prime numbers
 - how to find good random numbers
 - how to deal with “long” messages in, e.g., RSA: why we present operation modes in the context of symmetric ciphers and we do not cover them for asymmetric schemes?

HINT. Do not start from scratch for these issues, try to understand how a real-world implementation is handling these problems: some these problems are very common, hence see if the language libraries are providing an easy solution.

Homework #3: playing with PK schemes (4)

Resources:

- PKCS #1: RSA Cryptography Specifications Version 2.2 [\[RFC 8017\]](#)
- D. Boneh. Twenty Years of Attacks on the RSA Cryptosystem. [\[PDF\]](#)

Homework #3: playing with PK schemes (5)

Experimental comparison:

- real-world implementation of RSA: choose any well-known implementation, e.g., OpenSSL [rsa-util](#) or [PyCryptoDome](#)
- real-world implementation of AES: choose any well-known implementation, e.g., OpenSSL AES or [PyCryptoDome](#)

Evaluate the performance in encryption/decryption considering one file size (see constraints on input size from RSA!).

Homework #3: playing with PK schemes (5)

- many public implementations of RSA: it is ok to “look” at them to learn some details... but you should not do copy-and-paste code... **THE GOAL IS TO LEARN**
- I do not care for optimization: the goal of the performance comparison is to realize how fast or slow is an asymmetric scheme, such as RSA, compared to a symmetric scheme, such as AES.
- **NEVER USE YOUR RSA IMPLEMENTATION IN REAL-WORLD CODE**

Homework #4: Understanding ECC

Goals of the homework:

1. Understand the main ideas behind Elliptic-Curve Cryptography (ECC)
2. Understand the Elliptic-curve Diffie–Hellman (ECDH) key exchange protocol
3. Write a report discussing ECC and ECDH

Homework #4: Understanding ECC (2)

Useful resources:

- Lecture 16: Introduction to Elliptic Curves by Christof Paar [\[Video\]](#)
- Lecture 17: Elliptic Curve Cryptography (ECC) by Christof Paar [\[Video\]](#)
- Wikipedia: [\[ECC\]](#) [\[ECDH\]](#)
- A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography by Cloudfare [\[URL\]](#)

Homework #4: Understanding ECC (3)

The report should clearly define:

- **mathematical background:**
 - which group is generally used by ECC
 - how to perform a “computation” in this group, how to guarantee the properties from the group
 - order of the group, generator of the group
- **EC DLP**
- **ECDH**
- **(Optional) Double-and-Add Algorithm for Point Multiplication**

The report should be written for a non-expert user in cryptography, e.g., write the report to explain to one of your classmate what is ECC.

Homework #5: Understanding a real-world protocol



Two very well-known messaging applications are **WhatsApp** and **Telegram**. They use *custom* protocols to achieve specific security services (e.g., confidentiality).

Homework #5: Understanding a real-world protocol (2)

In this homework, you have to take a closer look at these two protocols. Your report can provide a discussion focused on different aspects:

1. A detailed discussion of one of two protocols: step-by-step discussion on how it works, what problems is trying to solve, why is not using other “simpler” solutions
2. A comparison of the two protocols: high level discussion of the two protocols and then a detailed discussion of the interesting differences and similarities
3. Historical discussion of a protocol: how it has been improved/changed over time, attacks that were revealed in the past, weaknesses and concerns
4. Play with reference/custom implementations, building a toy application using parts of the protocol.

You can choose one of these discussion directions or a mix of them.
If you have another proposal, ask me (publicly) on Piazza!

Homework #5: Understanding a real-world protocol (3)

Resources on WhatsApp/Signal:

- Wikipedia: [\[WhatsApp\]](#) [\[Signal Messenger\]](#) [\[Double Ratchet Algorithm\]](#)
- [WhatsApp Encryption Overview \(technical white paper\)](#)
- [Noise protocol](#)
- Paper [A Formal Security Analysis of the Signal Messaging Protocol](#)
- [Signal technical specifications and software libraries](#)
- Thesis [Security Analysis of the Signal Protocol](#)

Homework #5: Understanding a real-world protocol (4)

Resources on Telegram/MTProto:

- Wikipedia: [\[Telegram\]](#)
- [MTProto Mobile Protocol documentation](#) and [FAQs](#)
- [Telegram source code](#) (client)
- Google for “mtproto server implementation”, e.g., in GO there is [NebulaChat](#)
- Talk [Security Analysis of the Telegram IM](#) by Tomas Susanka at DEF CON 25
- Report [Security Analysis of Telegram](#)
- Paper [On the CCA \(in\)security of MTProto](#)
- Paper [Security Analysis of End-to-End Encryption in Telegram](#)