CSCI E-33a (Web50)
Section 8

Ref: Lecture 8 (Scalability and Security)

Vlad Popil

Nov 23, 2022

Welcome!

About me:

Volodymyr “Vlad” Popil

- 👨‍🏫 Teaching Fellow for E-33a (since 2019)
- 👨‍👧 Father of 4 y/o son
- 👨‍🎓 Master’s (ALM) in Software Engineering
- 👨‍💻 Software Engineer at Google
- 🇺🇦 Born in Ukraine

Email: vlad@cs50.harvard.edu

Sections: Wed 8:30-10:00 pm ET
Office Hours: Fri 7:00-8:30 pm ET

Agenda

- Logistics
- Lecture review
- Cryptography
- Certificates
- Hashing
- SQL Injection
- JS Attack
- **Final Project**
- **Heroku deployment (Django)**
- Grading criteria (reminders)
- **Tips**
- Q&A

Logistics

Intro

- Refer to website: https://cs50.harvard.edu/extension/web/2022/spring/
- Sections and office hours schedule on website sections
- Get comfortable with command line
- Text editor is usually sufficient to write code, BUT IDEs is faster!
- Zoom:
  - Use zoom features like raise hand, chat and other
  - Video presence is STRONGLY encouraged
  - Mute your line when not speaking (enable temporary unmute)
- 6 Projects
  - Start early (or even better RIGHT AWAY!!!)
  - Post **and answer** questions on Ed platform
  - Remember: bugs can take time to fix
  - Grade -> 3 × Correctness (5/5) + 2 × Design [code] (5/5) + 1 × Style [code] (5/5) (Project 0 is an exception)
  - E.g. 15+10+5=30/30 | e.g. Correctness can be 15, 12, 9, 6, 3, 0
  - Lateness policy - 0.1per minute => **16hrs 40 min**, plus one time 3-day extension
  - **FINAL PROJECT CAN'T BE LATE**
  - Set a reminder to submit the Google Form for each project
  - Final Project - Due Wednesday, Dec 11th at 11:59pm ET **<< 18 FULL DAYS LEFT >>**

10,000 foot overview

- *Section 0 - SKIPPED*
- *Section 1+2 (Git + Python)* - Chrome Dev Tools (Inspector), CDT (Network), Project 0, Grading aspects
- *Section 3 (Django)* - Env Config, Markdown, RegEx, IDEs, pycodestyle, Debugging, Project 1
- *Section 4 (SQL, Models, Migrations)* - VSCode, linting, DB modeling, Project 2
- *Section 5 (JavaScript)* - CDT + IDE's Debugging, Project 3
- *Section 6 (User Interfaces)* - Animations, cURL/Postman, jshint
- *Section 7 (Testing, CI/CD)* - **Project 4**, DB modeling, Pagination, Test Driven Development, DevOps
- *Section 8 (Scalability and Security)* - **Final Project**, Cryptography, CAs, Attacks, App Deployment (Heroku)

Burning Questions?

Please ask questions, or topics to cover today!

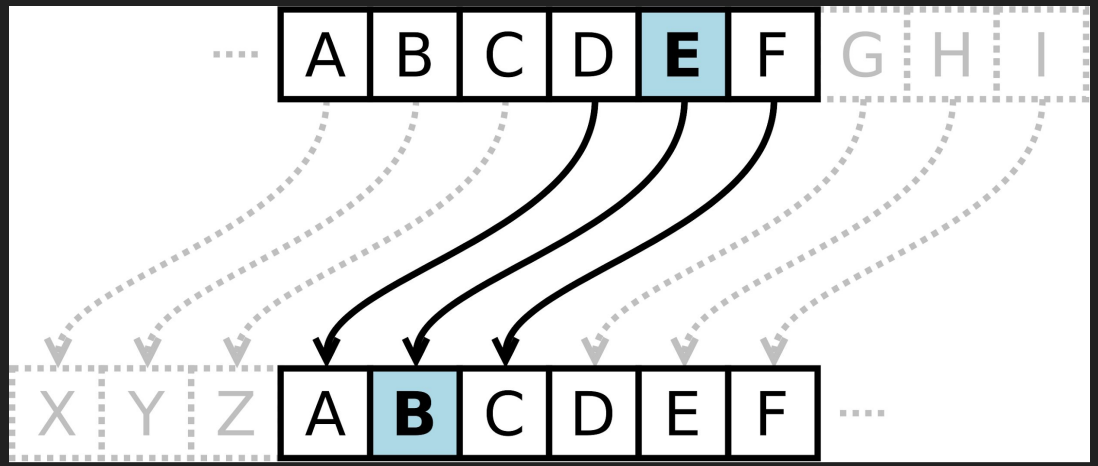
Topics:

- Proposals checked for enough scope?
  - Yes - comment in GradeScope, email, if scope is insufficient CONDITIONALLY APPROVE
  - No - sometime hard to judge, should be as much work as say project 2-4
- No option to submit final just yet.

Cryptography

Origins: Caesar Cipher

E.g python -> mvqelk

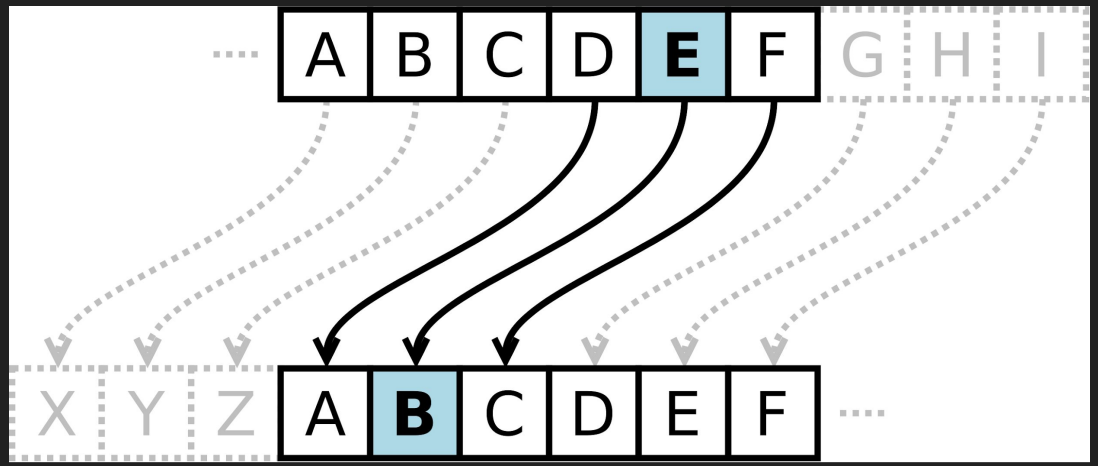


Step up: Vigenère cipher

Cryptography

Origins: Caesar Cipher

(100 B.C. – 44 B.C.)



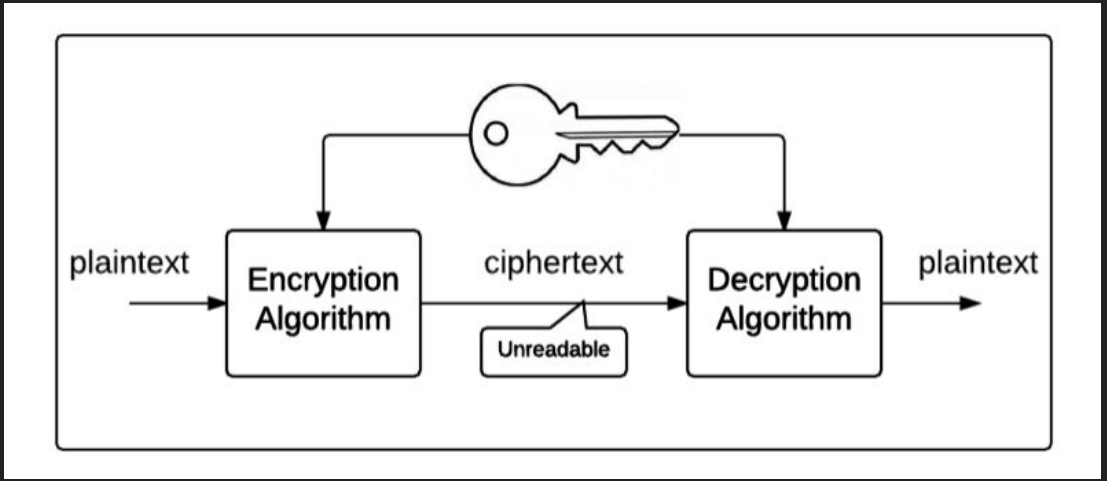
Step up: Vigenère cipher (1553)

First described in 1553, the cipher is easy to understand and implement, but it resisted all attempts to break it for three centuries until 1863.

Cryptography

Symmetric:

- DES
- AES
- 3DES
- ...

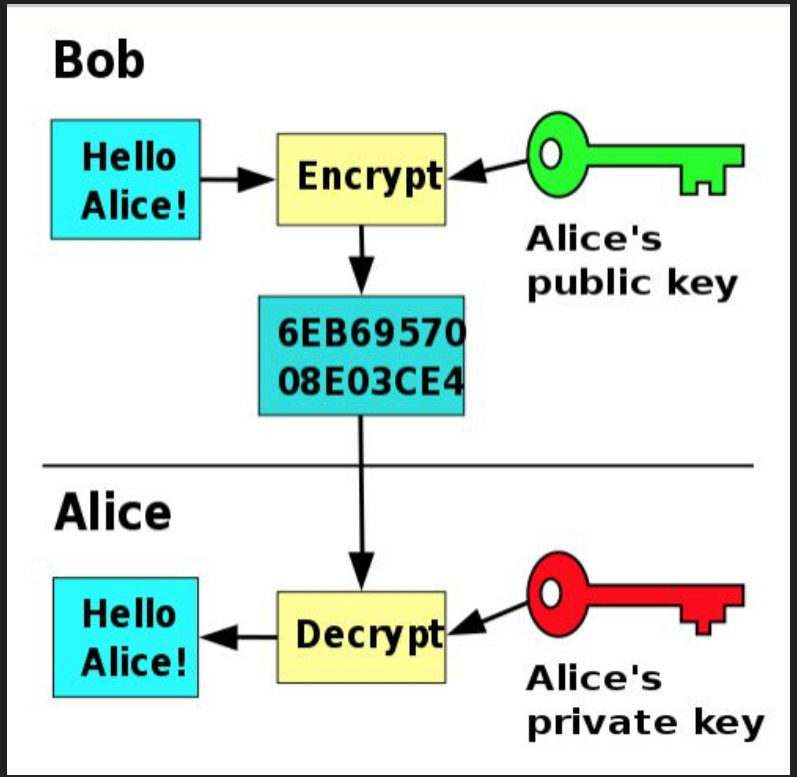


Cryptography

Asymmetric:

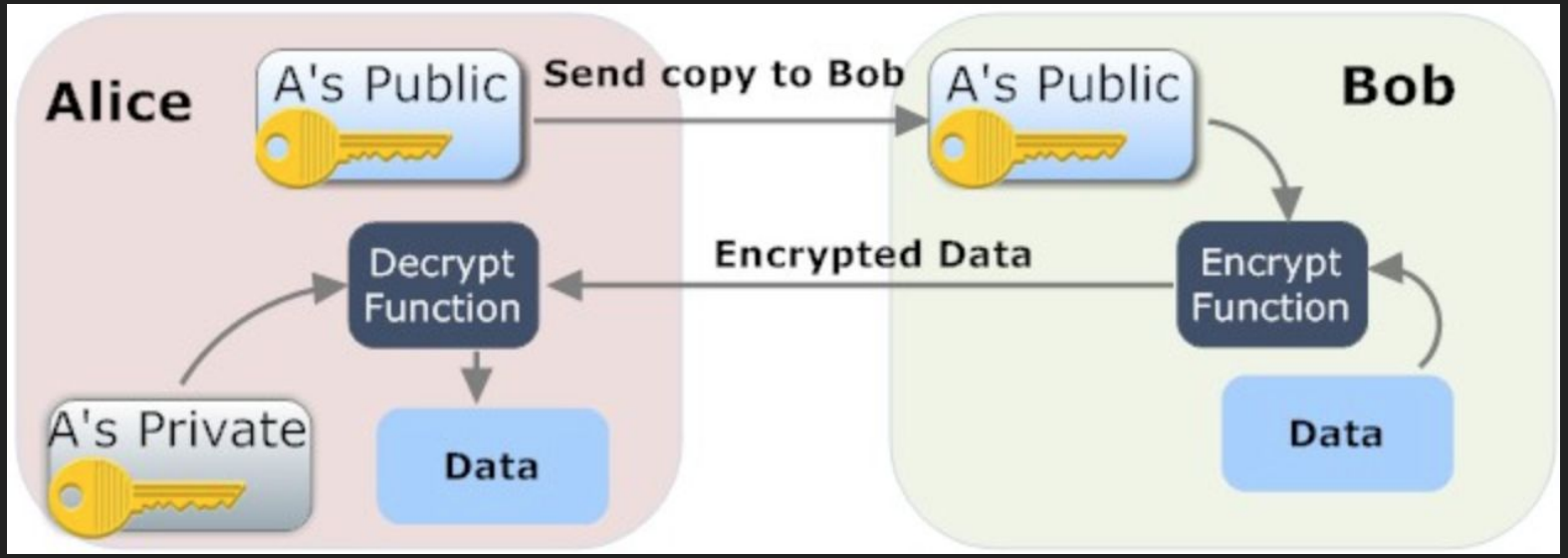
- RSA
- ElGamal
- Diffie-Hellman
- …

Video: https://www.youtube.com/watch?v=Py06YA4dfKE



Cryptography

Asymmetric (continued)

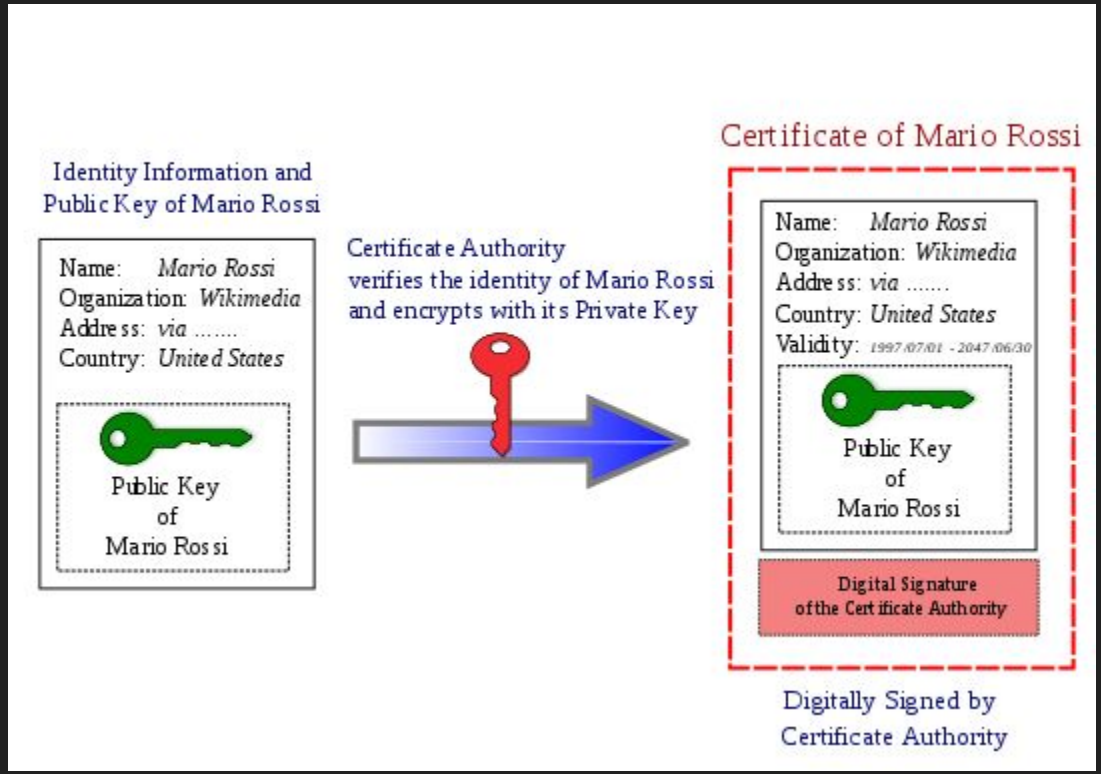


Demo

Let’s try something

1. I will share my public key in chat
2. Go to: https://www.devglan.com/online-tools/rsa-encryption-decryption
3. In RSA Encryption:
    a. In “Enter Plain Text to Encrypt” type something (appropriate).
    b. In “Enter Public/Private key” paste my public
    c. Click “Encrypt”
    d. Send back the value from “Encrypted Output (Base64)”
4. I’ll try to decode

Certificates



Hashing [+ real story]

password = ‘apple’

password_hashed =’3a7bd3e2360a3d29eea436fcfb7e44c735d117c42d1c1835420b6b9942dd4f1b’

Can we find out what is the password given hash?

Hashing with Salting

password = ‘apple’

salt = ‘Jfm2j3#-(%!2jgm*(#9n(#39jgfwefoii’

--

password_salted = ‘appleJfm2j3#-(%!2jgm*(#9n(#39jgfwefoii’ *(password+salt)*

password_salt_hash =’771fcaaa96c0be65023da9877d8012d0023c8e176782187c36ecf91c5b603b7f’

--

Can we find out what is the password given hash+salt?

Not likely

SQL Injection

user = execute("SELECT * FROM Users WHERE UserId = " + user_id)

INPUT [5; DROP TABLE Users]

Results -> SELECT * FROM Users WHERE UserId = 5; DROP TABLE Users;

JS Attack

Name: Samy Kamkar

Dare to click me?

Final Project

Final Project

Requirements:

1. Your web application must utilize **Django** (including at least **one model**) on the back-end and **JavaScript** on the front-end.
2. Your web application must be **mobile-responsive**.
3. In a README.md (whose extension can be .txt, .md, .adoc, or .pdf) in your project's main directory, include a short writeup describing your project, what's **contained in each file** you created, why you made certain design decisions, and any other additional information the staff should know about your project.
4. If you've added any Python packages that need to be installed in order to run your web application, be sure to add them to a **requirements.txt** file!
    a. Has to be working as `python -m pip install -r requirements.txt`

Final Project

Some suggestions:

- Start asap
- Provide sufficiently complex functionality
- Make sure functions properly → PROACTIVELY FIND AND FIX BUGS!!!
- Assure good design (pylint) - optionally
- Assure good style (pycodestyle/jshint)
- Suggestion (not graded):
  - Add Unit tests (incl. coverage assessment) + Selenium
  - Run with Docker (~)
  - Deploy to Heroku, etc.
  - GitHub Actions for automatic testing (at least)

Heroku

Deployment demo...

Heroku

- A cloud-based web-hosting platform
- Different from other cloud computing platforms in that it focuses on web hosting.
- Allows us to host Django applications (And many other applications) online
- Has a free option and several paid plans

Set up Django Project for Heroku.

1. In settings.py, replace you SECRET_KEY with an environment variable.
2. Set debug to False

Launch Application with Heroku

1. Create a Heroku Account
2. Install Heroku Command-Line interface
   a. Either click the link, or if you have homebrew installed: `brew install heroku/brew/heroku`
3. Type `heroku login` and follow instruction so log in
4. In the main directory of your Django project (which must be the only thing in the GitHub Repository!), add a requirements.txt file that includes the lines
   a. django
   b. gunicorn
   c. django-heroku
5. In that same directory, add a runtime.txt file with your current version of python (for me, python-3.7.2)

Launch Application with Heroku

6. type `heroku create myapp --buildpack heroku/python` to prepare Heroku to receive your source code. This will give you the url for your application and to a new github repository run by Django
7. Create a Procfile in the same directory and include in it: “web: gunicorn myproject.wsgi” (note that myproject is the name of your Django project, like commerce or wiki)
8. pip install django_heroku
9. In settings.py, include the line “import django_heroku” at the top and “**django_heroku.settings(locals())**” at the bottom
10. commit and push your changes to github:
    OR IF NEW PROJECT follow: https://devcenter.heroku.com/articles/git

Launch Application with Heroku

11. type “heroku config:set SECRET_KEY=some_value” into terminal to set up the secret key in your heroku configurations
12. Push to the new github repository by typing “git push heroku main”
13. type “heroku run python manage.py migrate” to migrate models (BETTER Procfile)
14. Log into your Heroku account and view your apps to see the website!
15. In the future:
    a. Use the online platform to pay for a custom domain name
        i. You can also choose to buy a domain name somewhere else and then add it
    b. in your terminal, type heroku logs to see requests and debug.

Design

What can be considered (not exclusively):

- Proper refactoring (copy-paste is usually a no-no)
- Use of constants
- Proper use of functions
- More reasonable solution
- Code/file structure
- Additional considerations: error preventions/handling
- Additional considerations for better application, username/password requirements, input data checks, security (hashing), etc.

Style

What can be considered (not exclusively):

- pycodestyle (indentations, line breaks, long lines)
- COMMENTS!
- Naming for variable, function, files, etc.
- Consistency is the key!

Random Tips

- Video Speed Controller (Chrome Extension)
- Spotify + Hulu + Showtime => $5
- GitHub Education Pack
- Windows licence (https://harvard.onthehub.com)
- Chrome Tabs → Marvelous Suspender extension
- DSA:
    - Video by CS50: https://www.youtube.com/watch?v=QDC1Ik-SeOI&ab_channel=CS50
    - Leetcode / AlgoExpert / Etc.
    - Stanford Algorithms Specialization (EdX link /Coursera) - more theory (time consuming)
    - e22 seems good!
    - e20 + e124 (combo) - HARD!
- System Design:
    - Grokking System Design
    - Alex Xu - System Design

Resources

- https://github.com/vpopil/e33a-sections-fall-2022

PLEASE FREE TO CONNECT

https://www.linkedin.com/in/vpopil

CSCI E-33a (Web50)
Section 8

Ref: Lecture 8 (Scalability and Security)

Vlad Popil

Nov 23, 2022

Good Bye!

and

Happy Thanksgiving