- **If you choose to do a presentation project:**
  - **Please email me 3 project topics you are interested in by Feb. 24th, indicating your first, second, and third choices. Please also indicate whether you want to present the two papers on the same day or on different days**
- **If you choose to do a systems project**
  - **Please email me the project topic by Feb. 24th**
- **If you choose to do a programming project**
  - **Please do NOT email me**

Presentation Projects

Presentation: April/May

Presentation slides submission: 11:59pm May 3

In the presentation project, you will present two papers using powerpoint slides. Each presentation takes about 25 minutes. You can choose to present the two papers on the same day or on different days. The presentation will be scheduled in April/May or at the end of March (You will receive an email from me at least 2 weeks before your presentation). The presentation slides should be uploaded on the blockboard by May 3.

1. Bitcoin blockchain developer guide (**Considered as two papers**)
https://bitcoin.org/en/developer-guide#block-chain

2. Introduction to Bitcoin and Blockchain **(Considered as two papers, nice tutorials on blockchain)**

http://chimera.labs.oreilly.com/books/1234000001802/ch02.html
http://chimera.labs.oreilly.com/books/1234000001802/ch05.html
http://chimera.labs.oreilly.com/books/1234000001802/ch07.html
http://chimera.labs.oreilly.com/books/1234000001802/ch08.html

3. Blockchain tutorial
https://www.edureka.co/blog/blockchain-tutorial/

3. A Blockchain-based Approach for Data Accountability and Provenance Tracking
https://arxiv.org/pdf/1706.04507.pdf

4. On the Security and performance of Proof of Work Blockchains

5. Using Blockchain and smart contracts for secure data provenance management
https://arxiv.org/pdf/1709.10000.pdf

6. ProvChain: A Blockchain-based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability

7. Security Implications of Blockchain Cloud with Analysis of Block Withholding Attack
https://www.cse.unr.edu/~toshd/assets/bwh-blockchain.pdf

8. Blockchain: A Graph Primer
https://arxiv.org/pdf/1708.08749.pdf

9. SCONE: Secure Linux Containers with Intel SGX
https://www.usenix.org/system/files/conference/osdi16/osdi16-arnautov.pdf

10. Challenges Towards Securing Hardware-assisted Execution Environments
http://www.cs.wayne.edu/fengwei/paper/tee-hasp17.pdf

11. *VC3*: Trustworthy Data Analytics in the Cloud using SGX
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/vc3-oakland2015.pdf

12. Foundations of Hardware-Based Attested Computation and Application to SGX
http://ieeexplore.ieee.org/abstract/document/7467358/

13. OpenSGX: An Open Platform for SGX Research
http://www.internetsociety.org/sites/default/files/blogs-media/opensgx-open-platform-sgx-research.pdf

14. SecureKeeper: Confidential ZooKeeper using Intel SGX
https://lsds.doc.ic.ac.uk/sites/default/files/securekeeper-middleware16.pdf

15. Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services
http://research.microsoft.com/pubs/160659/websso-final.pdf

16.Toward Black-Box Detection of Logic Flaws in Web Applications
http://www.internetsociety.org/sites/default/files/02_5_0.pdf

17.Detecting Logic Vulnerabilities in E-commerce Applications
http://www.internetsociety.org/sites/default/files/04_4_1.pdf

18. Vulnerability Statistics for e-banking System
http://www.ptsecurity.com/download/BankingWP.Fv4.pdf

19. Show Me the Money! Finding Flawed Implementations of Third-party In-app Payment in Android Apps
http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/ndss2017_05A-2_Yang_paper.pdf

Guideline (ppt slides)

- Motivation: E.g. Why role-based access control?
- Background: E.g. Syntax of xml
- Technical details
- Use enough examples/pictures to illustrate technical details.
- Related work

Guideline (presentation)

- Present slowly and clearly, and stop to ask if other students have any questions
- Do not directly read everything from slides
- Use examples to illustrate technical details

Programming Projects

No presentation is required
Submission deadline: 11:59pm, May 3 (Thur)
(source code + readme)

The programming project is done by a group of 2 students. You can use existing implementations of RSA and digital signature (e.g. provided in java.security, openssl, etc), if necessary. You are not required to implement a graphical user interface. If you work on the programming project by yourself, you will get 10 points extra credits.

This project implements a secure virtual election booth. The implementation should provide a secure way for people to vote online. The secure virtual election booth must meet the following requirements:

o No one can vote more than once.
o No one can determine the candidate for whom anyone else voted.

Assume that there are 3 voters (Alice, Bob, John) and 2 candidates (Tim and Linda). Each voter has a voter registration number. The voter registration number is given below, which is stored in a file **voterinfo**

Alice	**112550000**
Bob	**113880000**
John	**114660000**

The voters connect to the Voting Facility (VF) to vote. Assume that all voters have the public keys of VF and VF has the public keys of all voters. You can manually generate the public keys for VF and all voters, store them in files, and then used them in your program.

The client is invoked (by the voter) as:
voter-cli *<VF server's domain name> <VF server's port number> (C/C++)*
java VoterCli <VF server's domain> <VF server's port> (Java)
The VF server is invoked as:
vf *<VF server's port number> (C/C++)*
java Vf <VF server's port number> (Java)

CS558: The VF server must be a concurrent server that enables multiple voters to connect to the server simultaneously.
CS458: The VF server can be an iterative server

Let || represent concatenation, pub(X) represent the public-key of X, priv(X) represent the private-key of X, DS(X) represent the digital signature of X, and E(K,M) represent encrypting message M using key K. The detailed steps are given below:

1. The voter invokes *voter-cli* to connect to the VF server
2. Upon connection, the voter is prompted to enter his/her name and voter registration number *vnumber*
3. After the voter enters *vnumber*, voter-cli sends **E(pub(VF), name||vnumber)||DS(name)** to the VF server ("||" represents concatenation).
4. The VF server checks whether the name and vnumber received match the information in file **voterinfo**. If not, the VF server sends 0 to voter-cli. voter-cli then prints "Invalid name or registration number" and terminates the connection.
   Otherwise, VF sends 1 to voter-cli. Voter-cli prints the user's name and prompts the user to select an action to perform:

   Welcome, <user's name>
   Main Menu
   Please enter a number (1-4)
   1. Vote
   2. My vote history
   3. Election result
   4. Quit

5. If the voter enters "1", then voter-cli sends 1 to the VF server. The VF server checks whether the voter has voted (based on file ***history*** describe in Step 6). If so, VF sends 0 to voter-cli, and voter-cli prints "you have already voted" and displays the Main menu. Otherwise, VF sends 1 to voter-cli and voter-cli displays the following:

   Please enter a number (1-2)
   1. **Tim**
   2. **Linda**

6. After the user enters the number, the client sends the number to VF **encrypted using the public-key of VF**. The VF server then updates the result in file **"result"** which has the following Format (initially the total number of votes is 0):

    Tim <the total number of votes>
    Linda <the total number of votes>

   VF also adds the date and time when the voter votes to a file **"history"** that has the following format (if history does not exist, then create the file):

   <registration number> <date and time when the voter votes>

   If the user is the last user who voted (i.e. all users have voted), then the VF server prints the results using the following format:
   <Candidate's name> Win
   Tim <the total number of votes>
   Linda <the total number of votes>

   Vote-cli then displays Main Menu in step 4
   Otherwise, go to Main Menu in step 4.
7. If the user enters "2" (i.e. My vote history), the VF server retrieves the corresponding entry in file **"history"** and sends the entry to voter-cli. voter-cli then displays the entry to the user. Go to Main Menu in step 4
8. If the user enters "3" (i.e. election result), then the VF server checks whether all users have voted. If not, then VF sends 0 to voter-cli and voter-cli displays "the result is not available"

   Otherwise, VF sends voter-cli the candidate who wins the election and the number of votes each candidate got. Voter-cli then displays the results with the following format:

   <Candidate's name> Win
   Tim <the total number of votes>
   Linda <the total number of votes>

9. If the user enters "4", then voter-cli terminates.

Grading Guideline (CS558)
- Readme – 5'
- Makefile – 5'
- Encryption/decryption – 25'
- Concurrent server – 10'
- Other functionality: -- 55'

Grading Guideline (CS458)
- Readme – 5'
- Makefile – 5'
- Encryption/decryption – 25'
- Other functionality: -- 65'

Submission guideline

- If you use C/C++/Java, please hand in your **source code, all public and private keys,** and a **Makefile** electronically (**do not submit .o or executable code**). If you use C#, please handin your source code and executables. Please indicate whether the code compiles and runs correctly on bingsuns.binghamton.edu.
- Write **a README** file (text file, do not submit a .doc file) which contains
    - Your name and email address.

    - The programming language you use (C/C++/C#/Java)
    - Platform (Bingsuns/Linux)
    - How to execute your program.
    - Core code for encryption/decrption
    - Core code for implementing the concurrent server
    - (Optional) Anything special about your submission that the TA should take note of.
- Place all your files under one directory with a unique name (such as proj-[userid] for assignment 1, e.g. proj-pyang).
- Tar the contents of this directory using the following command.
  tar –cvf proj-[userid].tar [directory_name]
  E.g. tar -cvf proj-pyang.tar proj-pyang/
- Use the mycourses to upload the tared file you created above.

Systems Projects

Presentation + demo: May 1 (in class)
Submission deadline: 11:59pm May 3
(source code + slides)

Each project is done by a group of 2 students. Each group will give a 20-25 min presentation and show demo (if applicable) in the class on May 3. If you choose to do the project by yourself, you will get 10 points extra credits. You will also need to submit your code and presentation slides (if applicable) by 11:59pm on May 3. You may want to talk to me before you start working on the project.

1. Buffer Overflow Attack (language: C)

Work through the shell example given in
http://www.maths.leeds.ac.uk/~read/bofs.html
Note: Use a linux machine (instead of bingsuns) to do this project.

2. Kernel Rootkit II (lauguage: C)

Design and implement a linux kernel rootkit that modifies the system call table to hide the file you create (ls).

3. Virus (language: C)

Design and implement a virus that can infect all executable files under a specific directory. Note that the program can still execute after the program is infected with the virus. When the infected program executes, the virus will execute first, and then the program.

4.Online secure checkout systems

You can choose one of the following:

(1) understand the implement an online secure checkout system at your choice and give a presentation, or

(2) Use paypal express checkout integration (https://developer.paypal.com/docs/integration/direct/express-checkout/integration-jsv4/) to develop an online checkout system

5. Blockchain platforms

In this project, you will use and compare two opensource blockchain platforms at your choice. You will need to demonstrate how to use those blockchain systems and give a presentation to compare the two blockchain systems.