TUTORIAL ON CRYPTOGRAPHY

Kenneth Emeka Odoh

Created: July, 2018
Updated: N/A

Vancouver, BC

Version 1

Sunday, September 9, 2018, 6:00 PM

Study Group: Applied Cryptography For Engineers

Hosted by Kenneth Emeka Odoh

Our goal is to learn the fundamentals of modern cryptography. A solid foundation in cryptography is important for software developers that want to design secure modern computer systems. If you are interested in topics such as Distributed Systems, Cloud Computing, Blockchain and of course, Security, this is the study group for you. Our method will be to follow the book "A Graduate Course in Applied Cryptography" by Dan

Organizer tools

1 going

7 Gate Venture Labs
68 Water Street, #401 · Vancouver, BC

Recommended Textbooks

- A Graduate Course in Applied Cryptography by Dan Boneh and Victor Shoup, { https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf }.

- Hacking Secret Ciphers with Python by Al Sweigart,

  { https://inventwithpython.com/hackingciphers.pdf }.

- List of practical exercises { https://github.com/sodium-friends/learntocrypto }.

- http://math.tut.fi/~ruohonen/MC.pdf

Bio

Education

- Masters in Computer Science (University of Regina) 2013 - 2016
- A number of MOOCs in statistics, algorithm, and machine learning

Work

- Research assistant in the field of Visual Analytics 2013 - 2016
  - { https://scholar.google.com/citations?user=3G2ncgwAAAAJ&hl=en }
- Software Engineer in Vancouver, Canada 2017 -
- Regular speaker at a number of Vancouver AI meetups, organizer of distributed systems meetup, and Haskell study group and organizer of Applied Cryptography study group 2017 -

Contents

- Introduction
- Secret key cryptography
- Public key cryptography
- Protocols
- Privacy and anonymity
- Case studies
- Exercises

Introduction

Cryptology consist of the following fields.

- Cryptography
- Cryptanalysis

Cryptography is the process for encrypting and decrypting messages.

Cryptanalysis is the process of recovering plaintext from the cryptotext without the decryption key.

The holy grail of cryptography is to make cryptanalysis very **computationally infeasible**.

Stenography is the art of hiding message in medium that is not obvious. For example, hiding information in images {https://www.csmonitor.com/Science/2010/0630/How-Russian-spies-hid-secret-codes-in-online-photos}

Cryptography provides the following benefits

- Confidentiality
  - Information available to authorized entities
- Integrity
  - Protection from unauthorized information changes.
- Authenticity
  - It is possible to verify the identities of people, computer, and programs.

Cryptanalysis

- Frequency analysis
  - Kasiski method

- Hill method (plaintext - cryptotext attack)

Securing your Resources

- Begin with a **threat** model
  - Identify the adversaries that you are protecting against, if it is a serious adversary be ready for shock e.g heartbleed bug.
  - How long should the information be secure.
- Identify the **trust** model.
- Identify the computation resources for encrypting and decrypting with minimal bottleneck.
- Reduce attack surface. Identify all attack vectors and handle the cases.
- Reduce severity of breaches with careful design. This fall into **crisis response**.
- Cryptography should be used in the mix of other techniques e.g secure coding, access control (permissions management) among others.

Mathematical Foundation of Cryptography

- Factorization
  - Given an integer, n. Find all the prime, p that are factors of n.

- Discrete logarithm
  - Given a prime, p, and integer a and c. Find every s that satisfy $a^s = c \mod p$.

Quick primer on the mathematics of Cryptography. See {Page 1 - 28 / 95}

{ http://www.cs.helsinki.fi/u/karvi/advanced_1_12.pdf }

Secret Key versus Public key Cryptography

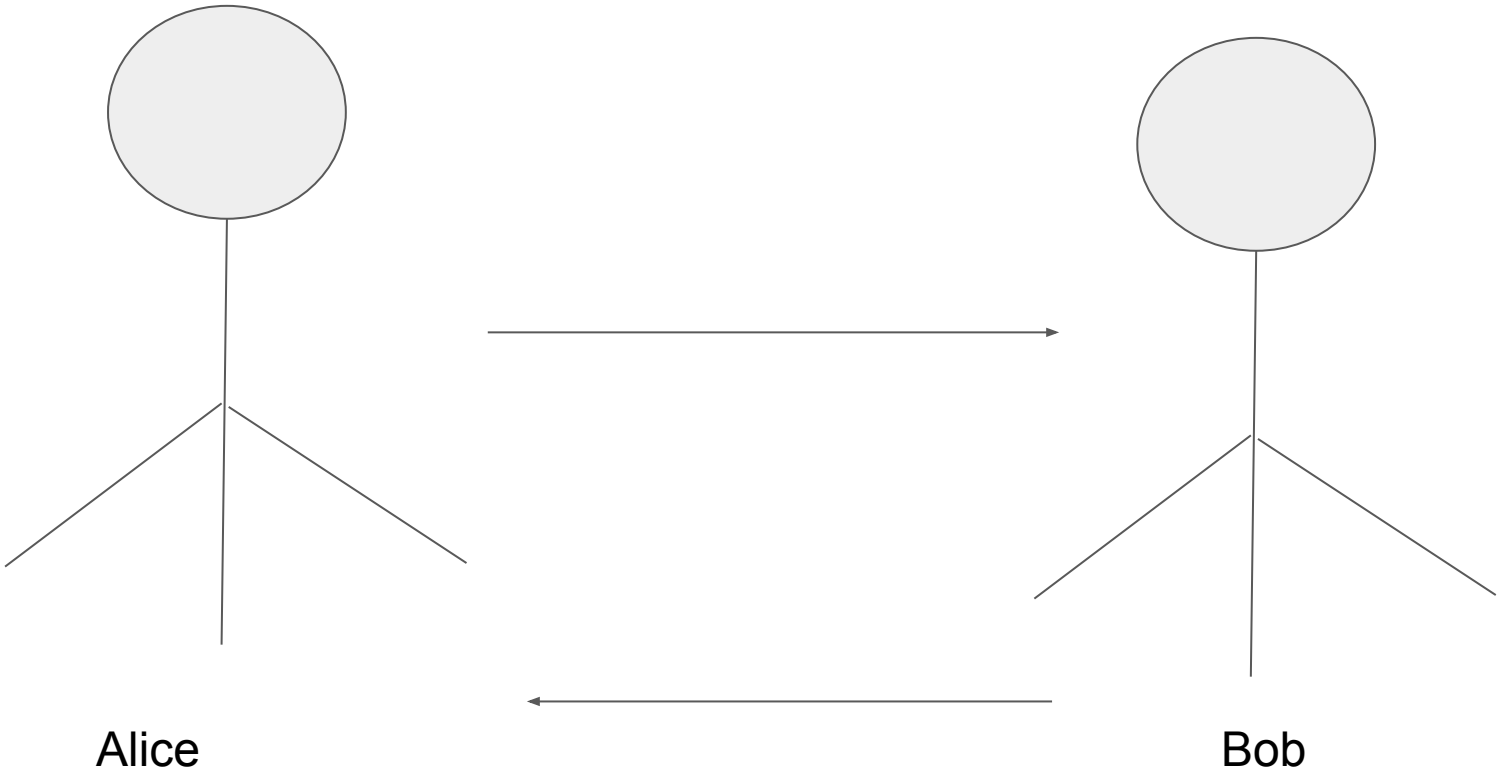

Figure 1: Communication between Alice and Bob

Secret key cryptography

The key cannot be made public without compromising security.

Public key cryptography

Each user has two keys (private key and public key), if is very difficult to get the private key from the public key. The public key can be announced with compromising security.

pt: plaintext

E_k: encryption using the key, k

D_k: decryption using the key, k

ct: cryptotext

$ct = E_k(pt)$

$pt = D_k(ct)$

$D_k(E_k (pt)) = pt$

Public key

Alice (n_A,e_A)

Bob (n_B,e_B)

Private key

Alice (n_A,d_A)

Bob (n_B,d_B)

Trapdoor vs hash function

- Trapdoor is a function that maps an input to an output. This is easy in the forward step from input to output but computationally expensive to recreate the input from the output. This is required for encryption and decryption function.

  1 = d*e mod n where d is an inverse of e

- Hash function is a one way function from an input to an output. It is impossible to recreate the input from the output.

{ https://www.uow.edu.au/~jennie/CSCI971/hash1.pdf }
{ http://www.cs.nthu.edu.tw/~wkhon/algo08-tutorials/tutorial-hashing.pdf }

RSA { Page 57 / 95 }

Diffie-hellman key exchange { Page 80 - 82 / 95 }

Elliptic curve { Page 85 - 92 / 95 }

{ http://www.cs.helsinki.fi/u/karvi/advanced_1_12.pdf }

Message Signature

- Alice, A, wants to send a message, m, to Bob, B
- Create S_A as the hash of m.
- Alice encrypts the hashed message using her private key, dA

$$D_A (S_A) = S_A^{dA} \bmod n_A$$

- Send message with signature as ($E_B(D_A (S_A))$, $E_B(m)$) to bob
- Once bob receives the message. He verifies the signature to be sure that there are no changes in the messages.

Kenneth Emeka Odoh

Protocols

Protocol is the sequence of communication steps between entities.

This describes the message format and position in the sequence for message delivery and receipt between the participating entities.

- Go to the Primer on protocol and discuss in details

  { https://www.cs.helsinki.fi/u/karvi/advanced_2_12.pdf }.

- Design principles for public key protocol {page 48 - 49 / 62}

- Discuss the design decisions of goldbug which a diverse collection of cryptographic cipher

  { https://compendio.github.io/goldbug-manual/ }

Privacy and Anonymity

Privacy vs Secrecy vs Anonymity

Secrecy is a **high degree** of **privacy**.
Anonymity is a way to achieve **privacy**.

- Is the information indecipherable to the unintended parties?
- It is possible to see the information, but can't tell where it is from?
- Who can see the sender, but cannot map to the real person?

  See more information on relationship between information provided and entropy

  - {https://www.johndcook.com/blog/2017/09/12/quantifying-the-information-content-of-personal-data/ }
  - {https://www.johndcook.com/blog/2017/09/20/quantifying-privacy-loss-in-a-statistical-database/ }

Zero-Knowledge method

This is a probabilistic proof of knowledge of an entity (**prover**), who claims to have information about a resource and proceeds to solve a set of challenges to verify the claim to the (**verifier**).

If the challenge is well designed, it is very difficult to perform **replay** attacks.

Discuss the examples

- Ali baba cave
- Colour-blind friend with a pair of balls

{https://en.wikipedia.org/wiki/Zero-knowledge_proof }

Properties of Zero-Knowledge proof

- **Completeness**: A fair verifier will be convinced by a honest prover that is if they are faithfully obeying the protocol.
- **Soundness**: it is only possible for the prover to cheat the verifier with a very small probability.
- **Zero-knowledge**: If the statement given by the prover is true, then the only information learnt by the verifier is that the statement of the challenge is true.

Mathematical basis for Zero-Knowledge proof

- Discrete logarithm
- Polynomial factorization
- NP-hard problem
    - Hamiltonian cycle

Discrete Logarithm

- **P** want to prove to **V** that it has knows the value of x.
- The following information is known to both **P** and **V** which consist of a value, y, prime, p, and generator, g.
- **P** creates y using her secret knowledge, x and sends y to **V**.
  - $y = g^x \bmod p$
- The knowledge of x can be used as a proof of identity to **V**.
- At a later time, On each round
- **P** generates a random number, r to compute C and send to **V**. Both options may be used only if the prover doesn't know the value of x and is trying to fool the verifier. This necessitates the need for Option 2 to validate identity of **P.**
  - Option 1

    $C = g^r \bmod p$
  - Option 2

    $C = g^{r+x \bmod (p-1)} \bmod p$
- On reception of message, C by **V.** It makes a random choice of either option 1 or 2.
- The verifier can however validate the prover's identity, **P** without knowledge of x.

Hamiltonian Cycle

- The graph, G is known to both **P** and **V**. Only **P** knows the hamiltonian cycle in the graph, G.
- At the beginning of each round, **P** creates H which is isomorphic to G.
- The knowledge of hamiltonian can be used as a proof of identity to **V**.
- At a later time, On each round.
- **P** creates a graph, H and commits using a commitment scheme to prevent changing of mind after sending H to **V**.
    - Option 1

      C = isomorphic permutation function between H and G
    - Option 2

      C = hamiltonian cycle in H
- On reception of message, C by **V.** It makes a random choice of either option 1 or 2.
- Verify isomorphic permutation function and hamiltonian cycle of H as a proof of identity of **P.**

Kenneth Emeka Odoh

zkSNARKs

This is similar to other concepts discuss in this course. However, it seem complicated to be discussed here for more information. This is used in **blockchain** applications.

For more details

- { https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/ }
- { https://getmonero.org/resources/moneropedia/ringsignatures.html }
- { https://getmonero.org/resources/moneropedia/stealthaddress.html }

Most of the zero-knowledge proof in this presentation require “trusted setup”. There are recent works that do not require “trusted setup”.

{https://zcoin.io/zcoin-moving-beyond-trusted-setup-in-zerocoin/ }

Trust setup is a case where a party is expected to create and delete the secret in the session.

Case Studies

Bitcoin: A simplistic view

- Bitcoin is a virtual currency that keeps a record of every transaction in a distributed append only public ledger known as blockchain.
- Transactions are securely processed and verified by using a proof of work consensus protocol with a reward scheme for miners.
- Bitcoin is a peer to peer network without a trusted central authority.
- An owner has full ownership over their money (electronic coins). It is possible to spend and receive increased or decreased value of coins without a central authority.

- Users have public and private key for secure communication.
- Digital signatures are added to message to prevent tampering and preventing fraud.
- Each user can have multiple public keys and private keys. Each of keys can be related to a wallet.
- Each user has a destination address which is obtained by hashing the public key provides **anonymity**.
- Bitcoin transaction is verified by a group of miners.
- Miners wait on a block in a queue like manner. A miner processes the transaction and waits on a quorum of users to verify the transaction.

Exercises

Code Documentation & Tutorial

- Read documentation for setting up node.js.
  - https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-16-04#how-to-install-using-nvm
  - https://www.w3schools.com/nodejs/default.asp
- Buffer { https://www.w3schools.com/nodejs/ref_buffer.asp }
- File management in Javascript
  - https://stackoverflow.com/questions/2496710/writing-files-in-node-js
  - https://stackoverflow.com/questions/14430859/javascript-best-way-to-convert-associative-array-to-string-and-back-the-other-w
- Peer to peer JavaScript
  - { https://github.com/socketio/socket.io-p2p }

- Install the following packages

```
npm install sodium-native
npm install duplex-json-stream
npm install net
```

- Get list of installed packages in node.js

```
npm list -g --depth=0
```

Required Exercises

- Coding up the entire exercises
  - { https://github.com/sodium-friends/learntocrypto/tree/master/problems }.

- Describe PGP (pretty good privacy) which uses elgamal algorithm

Optional Project

- Implement simplistic DSA without any complicated padding, message blinding, or protection against side channel attacks.
  - Implement DES
    - {http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm }
  - Decrypting DES
    - {https://crypto.stackexchange.com/questions/9674/how-does-des-decryption-work-is-it-the-same-as-encryption-or-the-reverse }

Advanced Project

- Implement the noise protocol as described in { https://noiseprotocol.org/noise.html }.

  - Hint: understand this block of code for peer to peer communication {https://github.com/socketio/socket.io-p2p/tree/master/examples/chat }

  - For more information on key generation for conference protocol {http://www.cs.helsinki.fi/u/karvi/advanced_3_12.pdf }

Conclusions

- Avoid implementing from scratch
  - Use existing library written by experts
- Side-channel attacks
- Rubber-hose cryptanalysis
- Quantum cryptography

Thanks for listening

https://github.com/kenluck2001?tab=repositories

@kenluck2001

https://www.linkedin.com/in/kenluck2001/

kenneth.odoh@gmail.com