Massachusetts Institute of Technology
6.857: Network and Computer Security
Professors Ronald L. Rivest and Yael Tauman Kalai

Handout 4
March 2, 2020
**Due:** March 23, 2020

# Problem Set 3

This problem set is due on *Monday, March 23, 2020* at **11:59 PM**. Please note our late submission penalty policy in the course information handout. Please submit your problem set, in PDF format, on Gradescope. *Each problem should be in a separate PDF.* When submitting the problem in Gradescope, ensure that **all your group members are listed on Gradescope**, and not in the PDF alone.

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza or email `6.857-tas@mit.edu`. You don't have to tell us your group members, just make sure you indicate them on Gradescope. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration.

*Homework must be submitted electronically!* Each problem answer must be provided as a separate pdf. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for LaTeX and Microsoft Word on the course website (see the *Resources* page).

**Grading:** All problems are worth 10 points.

With the authors' permission, we may distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this in your profile on your homework submission.

*Our department is collecting statistics on how much time students are spending on psets, etc. For each problem, please give your estimate of the number of person-hours your team spent on that problem.*

### Problem 3-1. Finite Fields and Groups

**(a) Galois Field 256**. We saw in lecture and recitation that AES uses the finite field $\mathrm{GF}(2^8)$ defined by the polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$. Note that $m(x)$ is irreducible. You may use any programming language to help you with the following parts. The parts assume the use of $GF(2^8)$ as used in AES, defined by $m(x)$.

1. Calculate the inverse of $x^5 + x^4 + x^3 + x^2$.
2. Calculate the discrete log of $x^7 + x^6 + x^2$ with respect to the generator $x^4 + x^2 + x + 1$.
3. Given some $a \in \mathrm{GF}(2^8)$, is it always possible to solve the equation $a = b^5$? What about the equation $a = b^4$? Explain your answer.

**(b) DDH Assumption**. Let $g$ be a generator of a group $G$ with order $t$. The DDH assumption states that, given $g^x$ and $g^y$, it is hard to distinguish between $g^{xy}$ and $g^u$, where $u$ is drawn uniformly at random from the set $\{1, \ldots, t\}$. Show that the DDH assumption is **false** for $\mathbb{Z}_p^*$.

**(c) Quadratic Residues**. Describe an algorithm that given $k$ outputs a $k$-bit prime $p$ and a generator for $\mathbb{Z}_p^*$.

### Problem 3-2. Encrypting With Paillier

In the lectures, we've seen ElGamal and RSA encryption schemes.

**(a)** Show that ElGamal and text-book RSA are homomorphic over the operation of multiplication. Namely, given pk, $\mathrm{Enc}(\mathrm{pk}, m_1)$, $\mathrm{Enc}(\mathrm{pk}, m_2)$ one can efficiently compute $\mathrm{Enc}(\mathrm{pk}, m_1 \cdot m_2)$, where in ElGamal multiplication is  mod $p$ and in RSA multiplication is  mod $n$.

The Paillier encryption scheme is another commonly used cryptosystem. It is homomorphic over the operation of addition rather than multiplication. Namely, given pk, $\text{Enc}(\text{pk}, m_1)$, $\text{Enc}(\text{pk}, m_2)$ one can efficiently compute $\text{Enc}(\text{pk}, m_1 + m_2)$, where addition is $\mod n$.

The Paillier cryptosystem is defined by the following 3 algorithms: $(\text{Gen}, \text{Enc}, \text{Dec})$:

- Gen takes as input a security parameter $1^\lambda$. Similar to RSA, it chooses two random prime numbers $p, q \in \{0,1\}^\lambda$ with leading bit equal to 1, and outputs $n := p \cdot q$ as the public key, and the pair $(a, b) := (\varphi(n), \varphi(n)^{-1} \mod n)$ as the secret key, where $\varphi(n) := (p-1) \cdot (q-1)$.
- Enc takes as input a public key $\text{pk} = n$ and a message $m \in \mathbb{Z}_n$. It chooses a random $r \in \mathbb{Z}_n^*$ and outputs the ciphertext $c = g^m * r^n \mod n^2$, where $g = n + 1$.
- Dec takes as input a secret key $(a, b)$ and a ciphertext $c \in \mathbb{Z}_{n^2}^*$ and outputs

$$m = \frac{(c^a - 1) \cdot b \mod n^2}{n}$$

  where $\frac{x}{y}$ is traditional division over the integers.

**(b)** Show that the Paillier encryption scheme is homomorphic over the operation of addition $\mod n$.

**(c)** In Paillier_decrypt.py, implement the decrypt function for the Paillier scheme. Do not change the parameters of the function. Include your completed code into your PDF submission.

**(d)** Decrypt the ciphertext found in Paillier_info.txt (along with key information). Hint: in python, use $pow(x, y, z)$ to do efficient modular exponentiation $x^y \mod z$. Additionally, use the // operator for the final division to ensure the output is an integer.

**Problem 3-3. Apple Find My Protocol**

In June 2019 at the WWDC 2019, Apple announced a new feature for iOS and OSX called Find My combining the features of Find My iPhone and Find My Friends into a single application. Summarized below is a high level description of the protocol.

1. With at least two Apple devices, the user's Apple devices create a shared private key communicated among them via end-to-end encryption.
2. Each pair of devices periodically creates a new secret key and public key pair using a deterministic algorithm applied to the previous secret key. This is also referred to as rotation of keys.
3. If Find My is enabled on a device, it emits its current public key via Bluetooth and other Apple devices nearby can pick up on the broadcast.
4. A device that has picked up a public key will then check its own location, encrypt the location using the public key, and upload this to Apple's servers along with a hash of the public key.
5. With a different Apple device, a user can then query Apple's servers with hashes of the public keys it has used to get the encrypted location associated with the hashed public key.
6. Using the original secret key, the user can decrypt and get the location of their lost device.

Read an article explaining the protocol by The Wired (`https://www.wired.com/story/apple-find-my-cryptography-bluetooth/`) and the description of the Find My protocol on Apple's Platform Security Manual (p. 103 - 105) (`https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf`), and answer the following questions. In the questions below, consider potential attackers to include someone who has stolen the user's device, other users who have Apple devices, and Apple employees with Find My database access.

**(a)** In step 2, why are the public keys periodically updated? What security properties are provided by updating the public key?

**(b)** For the encryption in step 4, what security properties must the encryption algorithm have for this to be secure? In particular, does it need to be CCA-secure or does CPA-security suffice?

**(c)** Can you come up with ways to attack the Find My protocol even assuming that the cryptographic primitives used in the protocol are secure? Some ideas worth considering are who are the trusted parties/ devices in the protocol, and who generates what data in this protocol.