MATH/CMSC 456 :: UPDATED COURSE INFO

Instructor: Gorjan Alagic (galagic@umd.edu)

Guest instructor: Carl Miller (camiller@umd.edu), ATL 3100K

Textbook: *Introduction to Modern Cryptography*, Katz and Lindell;

Homework 4 will be assigned next week, and due March 12.

Webpage: alagic.org/cmsc-456-cryptography-spring-2020/

Piazza: piazza.com/umd/spring2020/cmsc456

ELMS: active, slides and reading posted there.

Gradescope: active, access through ELMS.

TAs (Our spot: shared open area across from **AVW 4166**)

- Elijah Grubb (egrubb@cs.umd.edu) 11am-12pm TuTh (AVW);
- Justin Hontz (jhontz@terpmail.umd.edu) 1pm-2pm MW (AVW);

Additional help:

- Chen Bai (cbai1@terpmail.umd.edu) 3:30-5:30pm Tu (2115 ATL – inside **JQI**)
- Bibhusa Rawal (bibhusa@terpmail.umd.edu) 3:30-5:30pm Th (2115 ATL – inside **JQI**)

RECAP: Crypto and Arithmetic

Classical crypto is based on the hardness of certain computational problems.

We want to build cryptosystems out of hard arithmetic problems.

We studied the basics of **modular arithmetic.**

$\mathbb{Z}_q$ = the set of remainders mod q.

Arithmetic is performed on $\mathbb{Z}_q$ by always taking remainder mod q.

We asserted that all of the following can be done **efficiently** in $\mathbb{Z}_q$:

- Addition
- Multiplication
- Computing multiplicative (and additive) inverses.

PLAN FOR TODAY

1. Exhibit some needed efficient algorithms for arithmetic (exponentiation, and Euclid's algorithm).
2. Study the behavior of the exponential function mod q.
3. Construct a toy cryptosystem.

Last time we talked about the hardness of the factoring problem:

$$n = p_1 \cdot p_2 \cdot p_3 \cdot \cdots \cdot p_r$$

We are going to base our toy cryptosystem on a different, closely related problem: inverting the exponential function mod n.

SOME EFFICIENT ALGORITHMS FOR ARITHMETIC IN $\mathbb{Z}_q$ (APPENDIX B)

MODULAR ARITHMETIC (Review)

Let $\mathbb{Z}_q$ denote the set.

$$\mathbb{Z}_q = \{0,1,2,3, \dots, q-1\}$$

For any $a, b \in \mathbb{Z}_q$, the elements

$$[(a+b) \bmod q]$$

$$[(a \cdot b) \bmod q]$$

are also elements of $\mathbb{Z}_q$.

For any $a \in \mathbb{Z}_q$ and $n \geq 0$, the integer

$$[a^n \bmod q]$$

is an element of $\mathbb{Z}_q$.

ALGORITHM FOR EXPONENTIATION

How do we compute $[a^n \bmod q]$?

Attempt #1:

Simply compute $[a^i \bmod q]$ for $i = 1,2,3, \ldots n$.

That takes time at least exponential in the length (# of bits) of n. Too long.

Attempt #2:

Compute $[a^i \bmod q]$ for $i = 1,2,3, \ldots$ until we encounter a repeat. Extrapolate.

That could take time exponential in the length of q. Also too long.

ALGORITHM FOR EXPONENTIATION

How do we compute $[a^n \bmod q]$?

Attempt #3:

Write n in base 2:

$$n = (b_r b_{r-1} b_{r-2} \dots b_1 b_0)_2$$

Compute (by repeated squaring):

$$[a^2 \bmod q], [a^4 \bmod q], [a^8 \bmod q], \dots, [a^{(2^r)} \bmod q]$$

Compute:

$$[a^{1 \cdot b_0} a^{2 \cdot b_1} a^{4 \cdot b_2} \cdot \dots \cdot a^{2^r b_r} \bmod q]$$
$$= [a^{1 \cdot b_0 + 2 \cdot b_1 + 4 \cdot b_2 + \dots + 2^r b_r} \bmod q]$$
$$= [a^n \bmod q]$$

This <u>is</u> efficient!

ALGORITHM FOR EXPONENTIATION

Example:

Let $a = 2, n = 73, q = 9$.

Then,

$$n = (1001001)_2$$

By squaring,

$$[a^2 \bmod q] = 4, [a^4 \bmod q] = 7, [a^8 \bmod q] = 4, [a^{16} \bmod q] = 7, \ldots$$

Compute:

$$\begin{gathered}[a^n \bmod q] \\ = [a^1 a^8 a^{64} \bmod q] \\ = [2 \cdot 4 \cdot 7 \bmod q] \\ = \mathbf{2}.\end{gathered}$$

ALGORITHM FOR MULTIPLICATIVE INVERSES

Suppose that $\gcd(a, q) = 1$. We wish to compute $a^{-1} \bmod \mathrm{q}$.

Example: $q = 23$, $a = 15$.

1. Write down the two (obvious) equations $a \cdot 1 + q \cdot 0 = a$ and $a \cdot 0 + q \cdot 1 = q$.

2. Subtract the smallest right-hand quantity from the 2nd-smallest right-hand quantity.

3. Repeat step 2 until we obtain $a \cdot x + q \cdot y = 1$. Then, $ax = 1 \bmod q$.

$$15 \cdot 1 + 23 \cdot 0 = 15$$
$$15 \cdot 0 + 23 \cdot 1 = 23$$
$$15 \cdot (-1) + 23 \cdot 1 = 8$$
$$15 \cdot (2) + 23 \cdot (-1) = 7$$
$$15 \cdot (-3) + 23 \cdot (2) = 1$$

Exercise: What is $[9^{-1} \bmod 23]$?

Answer: 18

Answer: 20(=23-3), is the multiplicative inverse of 15 mod 23.

EFFICIENT OPERATIONS MOD q

	Efficient to compute?	**Efficient to invert?**
Addition	YES	YES
Multiplication	YES	YES
Exponentiation	YES	**????**

Question: Given $[a^n \bmod q], n, q$, can we efficiently compute a?

If not … perhaps we can use exponentiation to build a cryptosystem!

THE BEHAVIOR OF THE EXPONENTIAL FUNCTION IN $\mathbb{Z}_q$

EXERCISE (no calculators)

What is the **period** of the sequence

$$[a^0 \bmod 11], [a^1 \bmod 11], [a^2 \bmod 11], \ldots ?$$

(Meaning, how often does it repeat?). Answer this for $a = 2,4,5,10$.

Observation: All positive integers satisfy $a^{10} = 1 \bmod 11$.

AN OBSERVATION

When we know that

$$[a^y \bmod q] = 1,$$

inverting the map $f(a) = [a^n \bmod q]$ **may** become easy.
If $m = n^{-1} \bmod \mathrm{y}$, then let $g(a) = [a^m \bmod q]$. Then,

$$g\big(f(a)\big) = a^{mn} = a^{by+1} = a^1 \ mod\ q$$

(for some positive integer b)!

When can we compute such a y?

PRIME MODULI

Recall that a positive integer n > 1 is **prime** if it has no factors other than 1 and itself.

Theorem: If q is prime, then for any $a \in \{1,2,\ldots,q-1\}$,

$$a^{q-1} = 1 \bmod q.$$

(Example: q=11 is prime!)

This is **Fermat's Little Theorem.** How can we prove this?

PRIME MODULI

Theorem: If q is prime, then for any $a \in \{1,2,\ldots,q-1\}$,

$$a^{q-1} = 1 \bmod q.$$

Lemma 1: The map $\mathrm{h}: \{1,2,\ldots,q-1\} \to \{1,2,\ldots,q-1\}$ given by

$$\mathrm{h}(x) = [ax \bmod q]$$

is one-to-one and onto.

Proof of Lemma 1: Since q is prime, $\gcd(a,q) = 1$. Thus, by a result from the previous lecture, a has a multiplicative inverse mod q.

Let $\mathrm{g}: \{1,2,\ldots,q-1\} \to \{1,2,\ldots,q-1\}$ be defined by

$$\mathrm{g}(x) = [a^{-1}x \bmod q].$$

Then $g(h(x)) = h(g(x)) = x$, and so we conclude that h is one-to-one and onto. □

PRIME MODULI

Theorem: If q is prime, then for any $a \in \{1,2,\ldots,q-1\}$,
$$a^{q-1} = 1 \bmod q.$$

Lemma 2: Suppose that $x, y \in \mathbb{Z}$ and $b \in \{1,2,\ldots,q-1\}$ are such that
$$bx = by \bmod q.$$
Then,
$$x = y \bmod q.$$

Proof of Lemma 2: Since q is prime, $\gcd(b,q) = 1$, and b has a multiplicative inverse mod q. We have
$$x = b^{-1}bx = b^{-1}by = y \bmod q,$$
as desired. □

PRIME MODULI

Theorem: If q is prime, then for any $a \in \{1,2,\ldots,q-1\}$,

$$a^{q-1} = 1 \bmod q.$$

Proof of Theorem: Letting h be the function from Lemma 1 (mult. by a), we have

$$\{1,2,\ldots,q-1\} = \{h(1), h(2),\ldots,h(q-1)\}.$$

Taking the products of both sets, we find that

$$\begin{aligned} 1 \cdot 2 \cdot \cdots \cdot (q-1) &= h(1) \cdot h(2) \cdot \cdots \cdot h(q-1) \\ &= (a)(2a)(3a)\cdots[(q-1)a] \\ &= a^{q-1} \cdot 1 \cdot 2 \cdot \cdots \cdot (q-1) \bmod q \end{aligned}$$

Applying Lemma 2,

$$1 = a^{q-1} \bmod q.$$

□

PRIME MODULI

Conclusion: If q is prime, then for any $a \in \{1,2,\dots,q-1\}$, the sequence

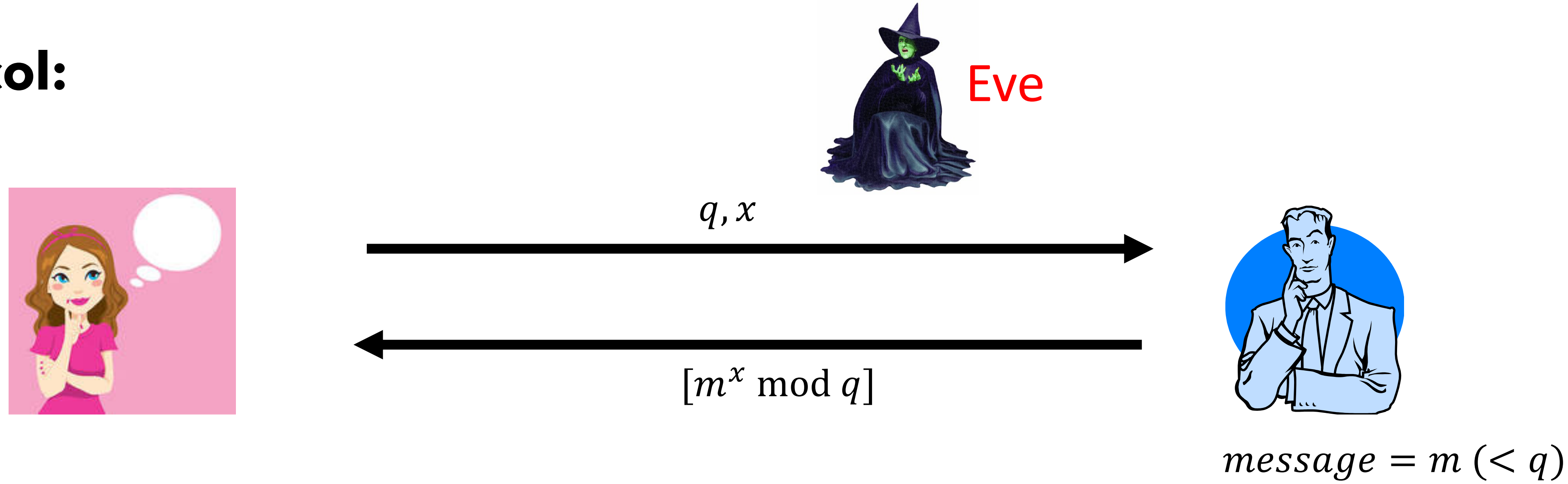
$$[a^0 \text{ mod q}], [a^1 \text{ mod q}], [a^2 \text{ mod q}], [a^3 \text{ mod q}], \dots$$

is periodic, and its period is a factor of $(q-1)$.

A TOY CRYPTOSYSTEM (PUBLIC-KEY ENCRYPTION)

FIRST ATTEMPT

Protocol:



1. Alice generates a random prime q and random $x \in \{1,2,\ldots,q-2\}$.
2. She computes $y = x^{-1} \bmod (q-1)$. (If it doesn't exist, start over.)
3. Bob transmits "ciphertext" $\mathrm{c} = [m^x \bmod q]$.
4. Alice computes "plaintext" $c^y = m^{xy} = m^1\, mod\; q$.

Problem: Alice can compute y, but so can anyone else!

NON-PRIME MODULUS?

Suppose that q is **not** prime.
We wish to prove an analogue of Fermat's Little Theorem.

Definition: Let $\phi(q)$ denote the **total number** of elements in $\{1,2,\ldots,q-1\}$ that have multiplicative inverses mod q.

Theorem: For any q>0 and any $a \in \{1,2,\ldots,q-1\}$,

$$a^{\phi(q)} = 1 \bmod q.$$

The proof is similar to the one for Fermat's Little Theorem.

Exercise: 13 and 17 are prime. What is $\phi(13 \cdot 17)$?

Answer: 192.

NON-PRIME MODULUS?

Consider the case $q = rs$, where r, s are prime.

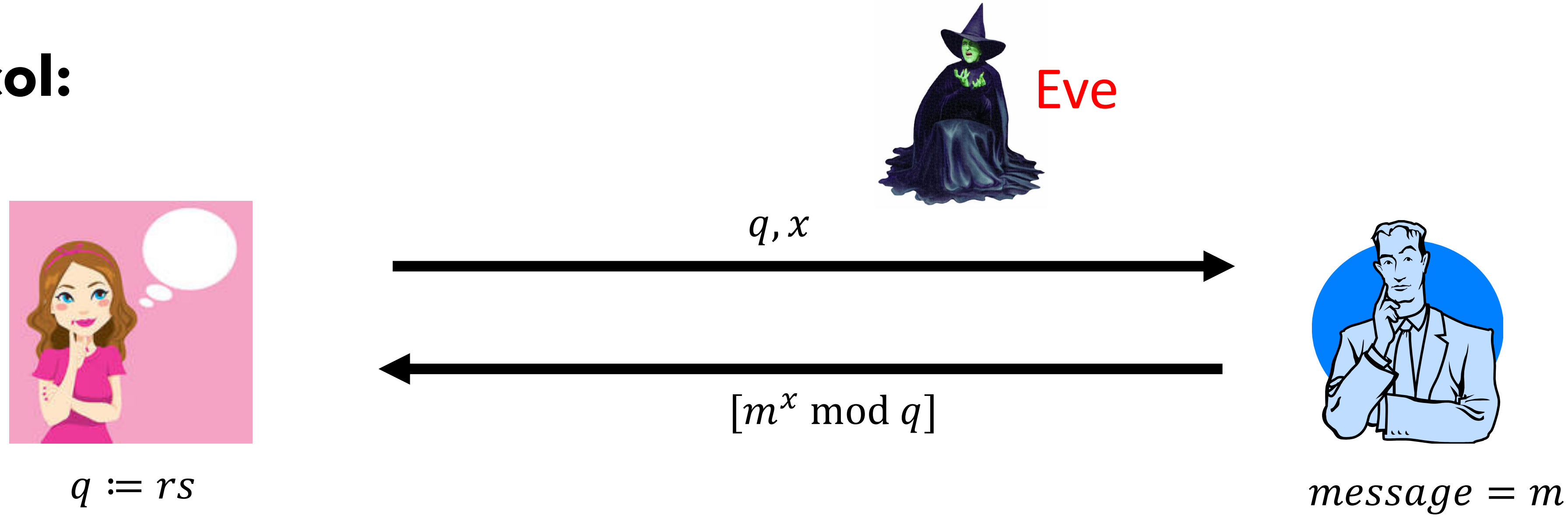
- There are $r - 1$ elements in $\{1,2, \dots, q - 1\}$ that are divisible by s.
- There are $s - 1$ elements in $\{1,2, \dots, q - 1\}$ that are divisible by r.
- There are 0 elements in $\{1,2, \dots, q - 1\}$ that are divisible by both.

Therefore,

$$\begin{aligned} \phi(q) &= \mathrm{rs} - 1 - (\mathrm{r} - 1) - (\mathrm{s} - 1) \\ &= (r - 1)(s - 1). \end{aligned}$$

A SECOND ATTEMPT

Protocol:



1. Alice generates random <u>primes</u> r,s and random $x \in \{1,2, \ldots, \phi(rs) - 1\}$.
2. She computes $y = x^{-1} \bmod \phi$(q), where $q = rs$. (If it doesn't exist, restart.)
3. Bob transmits ciphertext $c = [m^x \bmod q]$.
4. Alice computes "plaintext" $c^y = m^{xy} = m^1 \, mod \, q$.

Alice knows $\phi(q) = (r-1)(s-1)$. But there's no obvious way for Eve (who doesn't know r and s) to compute that quantity!

This (roughly speaking) is **<u>RSA encryption</u>.**

SUMMING UP

- We developed modular arithmetic some more (including an efficient algorithm for exponentiation).
- We stated **Fermat's Little Theorem** and discussed its implications for the inversion of the exponential function.
- We developed a toy version of **RSA encryption**, which is based on the hardness of inverting the exponential function.
- Next: We'll give a more formal treatment of public-key encryption and RSA.