Computer Science 4271
Spring 2023
Midterm exam 2 (solutions)
April 11th, 2023
Time Limit: 75 minutes, 11:15am-12:30pm

- Before starting the exam, you can fill out your name and other information of this page, but don't open the exam until you are directed to start. Don't put any of your answers on this page.
- This exam contains 6 pages (including this cover page) and 3 questions. Once we tell you to start, please check that no pages are missing.
- You may use any textbooks, notes, or printouts you wish during the exam, but you may not use any electronic devices: no calculators, smart phones, laptops, etc.
- You may ask clarifying questions of the instructor or TAs, but no communication with other students is allowed during the exam.
- Please read all questions carefully before answering them. Remember that we can only grade what you write on the exam, so it's in your interest to show your work and explain your thinking.
- By signing below you certify that you agree to follow the rules of the exam, and that the answers on this exam are your own work only.

The exam will end promptly at 12:30pm. Good luck!

Your name (print): ____________________

Your UMN email/X.500: ____________________`@umn.edu`

Number of rows ahead of you: __________ Number of seats to your left, to an aisle: __________

Sign and date: ____________________

Question	Points	Score
1	30	
2	30	
3	40	
Total:	100	

1. (30 points) Stream cipher mutability.

   Trusty Bank NA is a financial institution that sends transaction information over the Internet. They know they should protect the messages with cryptography, and they chose a high-quality stream cipher, but they forgot to do anything to protect the integrity of the messages.

   After playing with some sample transfer transactions, a criminal under the alias John Doe has managed to intercept an encrypted message that represents a transfer he initiated, reading "`Pay $1,000 to John`". He is considering different choices for modifying the message to transfer a larger amount of money to himself or one of his associates. In a table on the next page, we've shown the original message in characters (␣ represents a space) and in hexadecimal bytes, and the ciphertext John intercepted also in hexadecimal. Then there are 10 possible modified plaintexts that John might like the message to decode as, and 10 possible modified ciphertexts. Match the plaintexts with the ciphertexts.

   Note that while you may be able to get started just by looking at which characters have been modified, in some places you will need to use the mathematics of exactly how the stream cipher is computed.

   For your reference, here's a table between ASCII characters and their hex representations:

00 NUL	10 DLE	20	30 0	40 @	50 P	60 ‘	70 p
01 SOH	11 DC1	21 !	31 1	41 A	51 Q	61 a	71 q
02 STX	12 DC2	22 "	32 2	42 B	52 R	62 b	72 r
03 ETX	13 DC3	23 #	33 3	43 C	53 S	63 c	73 s
04 EOT	14 DC4	24 $	34 4	44 D	54 T	64 d	74 t
05 ENQ	15 NAK	25 %	35 5	45 E	55 U	65 e	75 u
06 ACK	16 SYN	26 &	36 6	46 F	56 V	66 f	76 v
07 BEL	17 ETB	27 ’	37 7	47 G	57 W	67 g	77 w
08 BS	18 CAN	28 (	38 8	48 H	58 X	68 h	78 x
09 HT	19 EM	29)	39 9	49 I	59 Y	69 i	79 y
0A LF	1A SUB	2A *	3A :	4A J	5A Z	6A j	7A z
0B VT	1B ESC	2B +	3B ;	4B K	5B [	6B k	7B {
0C FF	1C FS	2C ,	3C <	4C L	5C \	6C l	7C |
0D CR	1D GS	2D -	3D =	4D M	5D]	6D m	7D }
0E SO	1E RS	2E .	3E >	4E N	5E ^	6E n	7E ~
0F SI	1F US	2F /	3F ?	4F O	5F _	6F o	7F DEL

P	P	a	y	␣	$	1	,	0	0	0	␣	t	o	␣	J	o	h	n
PH	50	61	79	20	24	31	2c	30	30	30	20	74	6f	20	4a	6f	68	6e
CT	b8	9a	6b	c0	62	b6	fe	fd	a5	67	47	f7	a2	75	e4	80	c5	af
1.	P	a	y	␣	$	2	,	0	0	0	␣	t	o	␣	J	u	a	n
2.	P	a	y	␣	$	1	,	0	0	0	␣	t	o	␣	J	o	a	n
3.	P	a	y	␣	$	1	,	1	1	1	␣	t	o	␣	J	o	h	n
4.	P	a	y	␣	$	9	9	9	9	9	9	t	o	␣	J	o	h	n
5.	P	a	y	␣	$	1	9	0	0	0	␣	t	o	␣	J	o	h	n
6.	P	a	y	␣	$	5	,	0	0	0	␣	t	o	␣	J	o	h	n
7.	P	a	y	␣	$	9	,	0	0	0	␣	t	o	␣	J	o	h	n
8.	P	a	y	␣	$	1	0	0	0	0	␣	t	o	␣	J	o	h	n
9.	P	a	y	␣	$	1	0	,	0	0	0	␣	t	o	␣	J	o	n
10.	P	a	y	␣	$	2	,	0	0	0	␣	t	o	␣	J	o	h	n
(a)	b8	9a	6b	c0	62	b6	e2	fd	a5	67	47	f7	a2	75	e4	80	c5	af
(b)	b8	9a	6b	c0	62	b6	fe	fc	a4	66	47	f7	a2	75	e4	80	c5	af
(c)	b8	9a	6b	c0	62	be	fe	fd	a5	67	47	f7	a2	75	e4	80	c5	af
(d)	b8	9a	6b	c0	62	b5	fe	fd	a5	67	47	f7	a2	75	e4	9a	cc	af
(e)	b8	9a	6b	c0	62	be	eb	f4	ac	6e	5e	f7	a2	75	e4	80	c5	af
(f)	b8	9a	6b	c0	62	b5	fe	fd	a5	67	47	f7	a2	75	e4	80	c5	af
(g)	b8	9a	6b	c0	62	b6	e2	e1	a5	67	57	a3	b9	3a	8e	a5	c2	af
(h)	b8	9a	6b	c0	62	b6	eb	fd	a5	67	47	f7	a2	75	e4	80	c5	af
(i)	b8	9a	6b	c0	62	b6	fe	fd	a5	67	47	f7	a2	75	e4	80	cc	af
(j)	b8	9a	6b	c0	62	b2	fe	fd	a5	67	47	f7	a2	75	e4	80	c5	af

For each blank labeled with a ciphertext, write the number of the corresponding plaintext:

(a) __**8**__

(b) __**3**__

(c) __**7**__

(d) __**1**__

(e) __**4**__

(f) __**10**__

(g) __**9**__

(h) __**5**__

(i) __**2**__

(j) __**6**__

2. (30 points) Multiple choice. Each question after the first has only one correct answer: circle its letter.

   (a) The non-zero remainders mod 5, i.e. the numbers 1 through 4, form a group with the operation of multiplication mod 5. Which of them are generator(s) for the group? Circle all the answers that are generators.

   A. 1 **B. 2** **C. 3** D. 4

   The generators of a group are the elements where the set of all the powers of the element cover the entire group. Generators of groups like this (non-zero remainders modulo a prime) are used as the bases for exponentiation in discrete-log-based cryptography, though of course the numbers have to be much larger for security. Computing remainders mod 5 is relatively easy in decimal: you just look at the ones digit of a number, and subtract 5 if it is 5 or more. You can do the full exponent computation in regular integers, and then take the remainder, but there are several reasons why it is better to take the remainder after each multiplication: it keeps the sizes of the numbers smaller, and it makes it easier to see when you are in a repeating pattern.

   1 is never a generator in a group like this, because no matter how many times you multiply by itself, you only get 1 and no other values. If we consider 2, we can compute the first four powers of two as 1, 2, 4, and 8, which are congruent to 1, 2, 4, and $8 - 5 = 3$ *respectively. That covers all the elements of the group, so 2 is a generator. Similarly the first four powers of 3 are 1, 3, 9, and 27, which are congruent to 1, 3, 4, and 2, so 3 is also a generator. (Or if you compute* $3^3 \equiv 4$ *first,* $3 \cdot 4 = 12 \equiv 2$*.) For 4, if you compute the first four powers directly, they are 1, 4, 16, and 64, which are congruent to 1, 4, 1, and 4, respectively. You might guess from this that 1 and 4 will just alternate. That should also be clear in the version where you take the remainder at each step:* $4 \cdot 4 = 16 \equiv 1$*. So 2 and 3 are not covered and 4 is not a generator.*

   (b) This web security risk wasn't mentioned in lecture, but it did make it into the latest edition of the OWASP Top Ten:

   A. SSRF B. JavaScript injection C. XSCSS D. terminator cookies E. XXXRF

   Server-side request forgery, or SSRF, was number 10 on the 2021 OWASP Top Ten. None of the other answers are standard names web security risks: "JavaScript injection" would be a synonym for XSS, but is not widely used, and the others are just made up.

   (c) Which of these features indicates an insecure design for an interface used in creating unique temporary files?

   A. The caller can specify the directory to be used.
   B. The function chooses a unique filename, and the caller creates it.
   C. The generated filename includes pseudorandom data.
   D. The generated filename is based in part on the time.
   E. The generated filename includes the process ID.

   (d) Choose a pair of answers to fill in the two formulas in the following: A group of people are selecting items at random from a set with n items. The minimum number of selections to guarantee that at least one entry has been selected more than once is ____ , but the number that makes it more likely than not is about ____ .

   A. $n \ldots n/2$ B. $n^2 \ldots n$ C. $n \ldots n-1$ **D.** $\boldsymbol{n+1 \ldots \sqrt{n}}$ E. $n+1 \ldots \log n$

(e) If you have execute permission but not read permission on a Unix directory, this means that:

A. You can run programs in the directory via the PATH variable, but not by their names.

B. You can execute programs in the directory but not look at their code.

C. You can access files in the directory if you know their names.

D. When you run `ls` on the directory, it won't show you the types of files.

E. The execute permission has no effect on a directory.

(f) This library function is not a system call, but it needs to use system calls to do its job:

A. `sqrt` B. `strlen` C. `setuid` **D.** `system` E. `sprintf`

(g) Which of these lists of people gave their initials to a popular public-key cryptographic primitive?

A. Merkle, Damgård, and 5 other folks

B. Aaronson, Edwards, and Sipser

C. Rivest, Shamir, and Adleman

D. Simmons, Heninger, and Anderson

E. Diffie, Elgamal, and Schneier

(h) Revealing secret information a little bit at a time is a strategy used in many attacks. For instance we saw it as a way of revealing stack canary values, and in the vulnerable MAC in lab 10. Another place it is useful is:

A. Blind SQL injection

B. `$PATH` manipulation

C. Web crawling

D. Birthday attacks on hash functions

E. Filter-resistant cross-site scripting

(i) SQL injection and cross-site scripting are separate vulnerability types. But which of these is a kind of cross-site scripting that typically involves a database?

A. DOM-based cross-site scripting

B. Persistent cross-site scripting

C. Response splitting

D. Cross-site request forgery

E. Reflected cross-site scripting

3. (40 points) Matching definitions and concepts. Fill in each blank with the letter of the corresponding answer. Each answer is used exactly once.

(a) __**H**__ when a small input change could affect any part of the output

(b) __**L**__ when multiple names refer to the same inode

(c) __**R**__ a statement that is true in all situations

(d) __**C**__ a kernel abstraction that isolates all namespaces

(e) __**J**__ a system call fetching inode metadata from a file descriptor

(f) __**S**__ a kind of race between a check and an unsafe operation

(g) __**E**__ a declarative language for specifying document appearance

(h) __**O**__ if $h(x) = y$, another $x' \neq x$ with $h(x') = y$

(i) __**D**__ attack that submits a request on another web site

(j) __**T**__ an attack getting JavaScript to run under a different origin

(k) __**M**__ structure holding the permissions and contents of a file

(l) __**N**__ a security policy about preventing data modification

(m) __**B**__ a filesystem-only isolation technique

(n) __**G**__ a public-key anonymous key exchange protocol

(o) __**A**__ a system call to set a file's user and group

(p) __**P**__ a system call fetching inode metadata from a file name

(q) __**F**__ a mode of operation that can be used like a stream cipher

(r) __**I**__ if a quantum computer could solve this quickly, it could break RSA

(s) __**K**__ a UK government agency that develops cryptography

(t) __**Q**__ another word for the user with UID 0

A. `chown` B. `chroot` C. container D. CSRF E. CSS F. CTR G. Diffie-Hellman
H. diffusion I. factoring J. `fstat` K. GCHQ L. hard link M. inode N. integrity
O. second preimage P. `stat` Q. superuser R. tautology S. TOCTTOU T. XSS