CS 161 Spring 2025 — Introduction to Computer Security — Midterm

Name: _____

Student ID: _____

This exam is 110 minutes long.

Question:	1	2	3	4
Points:	0	12	18	20
Question:	5	6	7	Total
Points:	16	16	18	100

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (completely unfilled)
- ● Only one selected option (completely filled)
- ◍ Don't do this (it will be graded as incorrect)

For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares (completely filled)

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we may grade the worst interpretation.

Pre-exam activity (0 points):

Artwork by Anonymous

EvanBot here, EvanBot there, EvanBot everywhere!
Draw EvanBot from a different angle.

Q1 *Honor Code* (0 points)

I understand that I may not collaborate with anyone else on this exam, or cheat in any way. I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in, at minimum, negative points on the exam.

Read the honor code above and sign your name: _____

Q2 *True/False* **(12 points)**

Each true/false is worth one point.

Q2.1 The Caltopia Space Agency only allows a few critical employees to control a space shuttle's flight path, while the rest of the employees only get enough access to carry out their work.

TRUE or FALSE: This is an example of Least Privilege.

○ (A) TRUE ○ (B) FALSE

Q2.2 EvanBot designs a system that uses HMAC-DRBG with a truly random seed to generate secret keys used for symmetric encryption.

TRUE or FALSE: Based on Shannon's Maxim, we should assume that the attacker knows EvanBot is using HMAC-DRBG and can predict the generated secret key.

○ (A) TRUE ○ (B) FALSE

For the next two subparts: Suppose we have a little-endian C program with a local variable `char pancake[8]`. Consider the following GDB output after running the command `x/4wx pancake`:

```
0xfffd7014: 0xdeadbeef 0xffffffff 0xffff70ac 0x00000000
```

Q2.3 TRUE or FALSE: The value of `pancake[8]` is `0xff`.

○ (A) TRUE ○ (B) FALSE

Q2.4 TRUE or FALSE: The value of `pancake[0]` is `0xef`.

○ (A) TRUE ○ (B) FALSE

Q2.5 TRUE or FALSE: The first listed variable of a struct is stored at the lowest address.

○ (A) TRUE ○ (B) FALSE

Q2.6 TRUE or FALSE: During a function call in x86, arguments are pushed onto the stack in the order they appear in the function definition.

○ (A) TRUE ○ (B) FALSE

Q2.7 TRUE or FALSE: A buffer overflow vulnerability is impossible when stack canaries are enabled, because canaries protect the entire stack from arbitrary overwrites.

○ (A) TRUE ○ (B) FALSE

Q2.8 TRUE or FALSE: CBC mode encryption is IND-CPA secure even if the IV is reused across multiple encryptions with the same key.

○ (A) TRUE ○ (B) FALSE

Q2.9 TRUE or FALSE: RSA encryption without proper padding schemes (e.g., OAEP) is IND-CPA secure, provided the key size is sufficiently large.

○ (A) TRUE ○ (B) FALSE

Q2.10 True or False: Rollback resistance ensures that an attacker cannot guess the next generated bit in a pseudorandom number generator.

○ (A) True ○ (B) False

Q2.11 True or False: Public-key encryption is used in hybrid encryption because it can encrypt large amounts of data quickly.

○ (A) True ○ (B) False

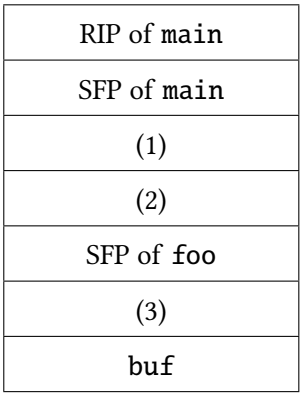
Q2.12 True or False: One-time pads are inconvenient because the keys can never be reused and need to be at least as long as the plaintext.

○ (A) True ○ (B) False

Q3 *Pigeons In the Coal Mines - Memory Safety* **(18 points)**

Consider the following vulnerable C code:

```
1  void foo() {
2      char buf[16];
3
4      fread(buf, 1, 16, stdin);
5      printf("%s", buf);
6      gets(buf);
7  }
8
9  int main() {
10     foo();
11     return 0;
12 }
```



Assumptions:

- Stack canaries are enabled, but no other memory safety defenses are enabled.
- You can use `SHELLCODE` as a 20-byte shellcode.
- You run GDB once and find that the address of `buf` is `0xffffffa0`.

Q3.1 (1 point) Fill the blanks in the stack diagram, assuming the program is paused on Line 3.

- ○ (A) (1) canary (2) `buf` (3) RIP of `foo`
- ○ (B) (1) canary (2) RIP of `foo` (3) canary
- ○ (C) (1) RIP of `foo` (2) canary (3) canary
- ○ (D) (1) canary (2) RIP of `foo` (3) SFP of `foo`

Q3.2 (1 point) What type of vulnerability is present in this code?

- ○ (A) Format string vulnerability
- ○ (B) Buffer overflow
- ○ (C) Signed/unsigned
- ○ (D) Off-by-one

In the next three subparts, provide an exploit that executes `SHELLCODE`.

Q3.3 (2 points) Give an input to `fread` on Line 4.

If a part of the input can be any non-zero value, use `"A"*n` to represent n bytes of garbage.

- ○ (A) `"A"*12 + "\xa0\xff\xff\xff"`
- ○ (B) `"A"*12 + "\xb8\xff\xff\xff"`
- ○ (C) `"A"*16`
- ○ (D) `"A"*15 + "\x00"`

Q3.4 (2 points) Let `OUTPUT` be the value printed by the program from the `printf` on Line 5. Which slice of `OUTPUT` gives the value of the stack canary, assuming you have the correct input to the previous subpart?

Note: For example, `[0:4]` means the first four bytes of `OUTPUT`.

- ○ (A) `[0:4]`
- ○ (B) `[4:8]`
- ○ (C) `[8:12]`
- ○ (D) `[12:16]`
- ○ (E) `[16:20]`
- ○ (F) `[20:24]`

Q3.5 (2 points) Let `CANARY` be the correct slice of `OUTPUT` from the previous subpart.

Give an input to `gets` on Line 6.

- ○ (A) `"A"*16 + CANARY + "A"*4 + "\xbc\xff\xff\xff" + SHELLCODE`
- ○ (B) `"A"*16 + CANARY + "A"*4 + "\xb8\xff\xff\xff" + SHELLCODE`
- ○ (C) `SHELLCODE + CANARY + "\xa0\xff\xff\xff"`
- ○ (D) `SHELLCODE + "A"*4 + CANARY + "\xa0\xff\xff\xff"`

Q3.6 (2 points) Which memory safety defenses, when enabled alongside stack canaries, would cause the correct exploit (without modifications) to fail? Consider each choice independently.

Note: For the PACs option only, assume the system is 64-bit (the exploit remains unchanged).

- ☐ (A) Pointer authentication codes
- ☐ (B) Non-executable pages
- ☐ (C) None of the above

For this rest of this question, **ASLR and stack canaries** are both enabled. In the next two subparts, provide an exploit that executes `SHELLCODE`.

Q3.7 (3 points) Give an input to `fread` on Line 4.

If a part of the input can be any non-zero value, use `"A"*n` to represent n bytes of garbage.

Q3.8 (5 points) Let `OUTPUT` be the output from the `printf` call on Line 5. You may slice this value (e.g. `OUTPUT[0:4]` returns the the first word of `buf`). You may also perform arithmetic on this value (e.g. `OUTPUT[0:4] - 7`) and assume it will be converted to/from the correct types automatically.

Also, let `CANARY` be the correct slice of `OUTPUT` from Q3.4.

Fill in each blank with an integer to provide an input to the `gets` call on Line 6.

Note that the `+` between terms refers to string concatenation (like in Project 1 syntax), but the minus sign in the second line refers to subtracting from the `OUTPUT[_:_]` value.

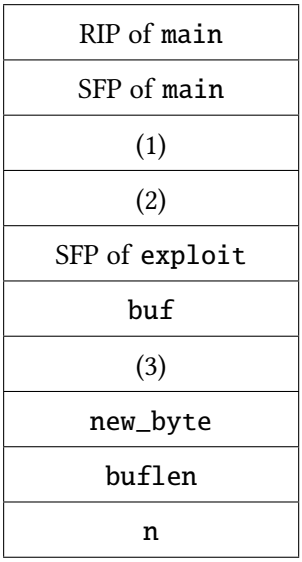
```
'A'*________ + CANARY + 'A'*________ +

(OUTPUT[______:______] - ________) + SHELLCODE
```

Q4 *ASLR (mod 5) - Memory Safety* **(20 points)**

Consider the following vulnerable C code:

```
1  void exploit() {
2      char buf[16];
3      size_t k = 0;
4
5      char new_byte = fgetc(stdin);
6      fgets(buf, 21, stdin);
7
8      size_t buflen = strlen(buf);
9      int n = 5;
10     while (n*k <= buflen) {
11         buf[n*k] = new_byte;
12         k += 1;
13     }
14 }
15
16 void sh_fn() {/* Code not shown */}
17
18 int main() {
19     // Function pointer
20     void (*shellcode)() = &sh_fn;
21     exploit();
22     return 0;
23 }
```



Non-executable pages are enabled. All other memory safety defenses are disabled.

This is the result of running `disas main` in GDB:

```
1  0x080760A0: push %ebp
2  0x080760A4: mov %esp, %ebp
3  0x080760A8: sub $4, %esp
4  ...
5  0x080760C8: call exploit
6  0x080760CC: mov $0, %eax
7  0x080760D0: add $4, %esp
8  0x080760D4: mov %ebp, %esp
9  0x080760D8: pop %ebp
10 0x080760DC: ret
```

Q4.1 (1 point) Fill in the blanks for the stack diagram, assuming the program is paused at Line 5.

- ○ (A) (1) `shellcode` (2) `buf` (3) RIP of `fgetc`
- ○ (B) (1) `shellcode` (2) RIP of `exploit` (3) `k`
- ○ (C) (1) `shellcode` (2) RIP of `fgetc` (3) SFP of `fgetc`
- ○ (D) (1) RIP of `exploit` (2) `k` (3) RIP of `fgetc`

Q4.2 (2 points) What is the value of the RIP of `exploit`, assuming the program is paused on Line 5?

- ○ (A) `0x080760A4`
- ○ (B) `0x080760C8`
- ○ (C) `0x080760CC`
- ○ (D) `0x080760D0`
- ○ (E) `0x080760D4`
- ○ (F) `0x080760DC`

In the next two subparts, provide an exploit that causes the program to execute `sh_fn`.

Q4.3 (3 points) Provide an input to the `fgetc` on Line 5.

- ○ (A) `0x00`
- ○ (B) `0xA0`
- ○ (C) `0xA4`
- ○ (D) `0xA8`
- ○ (E) `0xD0`
- ○ (F) `0xD4`
- ○ (G) `0xD8`
- ○ (H) `0xDC`

Q4.4 (3 points) Provide an input to the `fgets` on Line 6.

If a part of the input can be any non-zero value, use `"A"*n` to represent n bytes of garbage.

Q4.5 (3 points) How many different values of the variable `n` (on Line 9) (including `n = 5`) would result in the correct exploit, without modifications, executing `sh_fn`?

- ○ (A) 1
- ○ (B) 2
- ○ (C) 3
- ○ (D) 4
- ○ (E) 5
- ○ (F) 6
- ○ (G) 7
- ○ (H) 8

Q4.6 (2 points) Which memory safety defenses, when enabled alongside non-executable pages, would cause the correct exploit (without modifications) to fail? Consider each choice independently.

Note: For the PACs option only, assume the system is 64-bit (the exploit remains unchanged).

- ☐ (A) Pointer authentication codes
- ☐ (B) Stack canaries
- ☐ (C) None of the above

Q4.7 (3 points) Which modifications to the program itself would prevent the correct exploit, without modifications, from executing `sh_fn`?

Consider each choice independently.

- ☐ (A) Changing Line 6 to `fgets(buf, 17, stdin)`
- ☐ (B) Changing Line 8 to `int buflen = strlen(buf)`
- ☐ (C) Changing Line 10 to `while (n*k < buflen)`
- ☐ (D) Changing Line 12 to `k += 2`
- ☐ (E) None of the above

Q4.8 (3 points) In this subpart only, **assume ASLR is also enabled.** What is the approximate probability that the correct exploit, without modifications, executes `sh_fn`?

Clarification after exam: Assume ASLR randomizes the code section.

○ (A) 0 ○ (B) $\frac{1}{256}$ ○ (C) $\frac{1}{2}$ ○ (D) 1

Q5 *AES-COMBO - Symmetric Cryptography* **(16 points)**

EvanBot designs the AES-COMBO mode of operation, defined below:

$$C_1 = E_K(IV_1 \oplus P_1)$$
$$C_2 = E_K(IV_2 \oplus P_2) \oplus C_1$$
$$C_i = E_K(C_{i-2} \oplus P_i)$$

Q5.1 (1 point) Select the correct decryption formula for $i \geq 3$.

- ○ (A) $P_i = D_K(C_i \oplus C_{i-2})$
- ○ (B) $P_i = E_K(C_i) \oplus C_{i-1}$
- ○ (C) $P_i = D_K(C_i) \oplus C_{i-1}$
- ○ (D) $P_i = D_K(C_i) \oplus C_{i-2}$

Q5.2 (3 points) Select all methods for generating IV_1 and IV_2 that result in AES-COMBO being IND-CPA secure.

All choices are independent of each other.

- ☐ (A) IV_1 and IV_2 are independently randomly generated.
- ☐ (B) Seed a PRNG with K, set $IV_1 =$ `Generate(128)`, and then set $IV_2 =$ `Generate(128)` using the same PRNG instance.
- ☐ (C) Seed two separate PRNGs with K, set $IV_1 =$ `Generate(128)` from the first PRNG, and then set $IV_2 =$ `Generate(128)` from the second PRNG.
- ☐ (D) IV_1 is randomly generated and $IV_2 = H(IV_1)$.
- ☐ (E) IV_2 is randomly generated and $IV_1 = H(IV_2)$.
- ☐ (F) None of the above

In the next two subparts, suppose a ciphertext C gets modified in transit. Let P' represent the plaintext from decrypting C'. For each row, select the corresponding value. "Garbage" refers to a pseudorandom string, e.g. an unknown value decrypted with a block cipher.

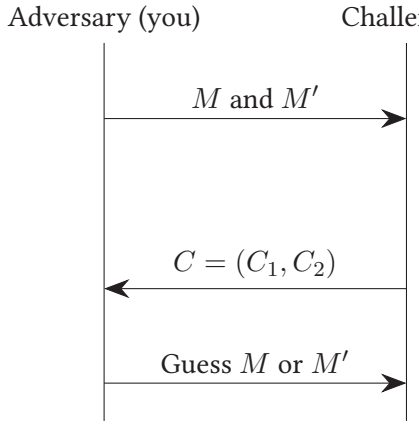
Q5.3 (3 points) C is modified such that $C'_1 = C_1 \oplus 1$ (i.e. a bit flip in the first ciphertext block).

P'_1:	○ (A) Garbage	○ (B) $P_1 \oplus 1$	○ (C) $P_1 \oplus P_2$	○ (D) P_1
P'_2:	○ (A) Garbage	○ (B) $P_2 \oplus 1$	○ (C) $P_2 \oplus P_1$	○ (D) P_2
$P'_i, i \geq 5, i$ **even**:	○ (A) Garbage	○ (B) $P_i \oplus 1$	○ (C) $P_i \oplus P_{i-1}$	○ (D) P_i
$P'_i, i \geq 5, i$ **odd**:	○ (A) Garbage	○ (B) $P_i \oplus 1$	○ (C) $P_i \oplus P_{i-1}$	○ (D) P_i

Q5.4 (3 points) C is modified such that $C'_2 = C_2 \oplus 1$.

P'_1:	○ (A) Garbage	○ (B) $P_1 \oplus 1$	○ (C) $P_1 \oplus P_2$	○ (D) P_1
P'_2:	○ (A) Garbage	○ (B) $P_2 \oplus 1$	○ (C) $P_2 \oplus P_1$	○ (D) P_2
$P'_i, i \geq 5, i$ **even**:	○ (A) Garbage	○ (B) $P_i \oplus 1$	○ (C) $P_i \oplus P_{i-1}$	○ (D) P_i
$P'_i, i \geq 5, i$ **odd**:	○ (A) Garbage	○ (B) $P_i \oplus 1$	○ (C) $P_i \oplus P_{i-1}$	○ (D) P_i

Assume for the following subparts only that $\mathbf{IV_1} = \mathbf{IV_2} = \mathbf{IV}$ for some randomly generated IV. You want to provide a strategy to win the IND-CPA game.



First, the adversary (that's you!) sends two different challenge messages, $M \neq M'$, to the challenger. For your strategy, you can assume M and M' are each **two blocks long**.

Then, the challenger randomly encrypts either M or M'. The resulting two-block ciphertext $C = (IV, C_1, C_2)$ is returned to you.

Finally, you guess whether M or M' was encrypted.

NOTE: The diagram originally had a typo with $C = (C_0, C_1)$.

In this strategy, the query phase is not needed (i.e. you never have to ask the challenger to encrypt messages of your choosing beforehand).

The second challenge message $M' = (?, ?)$ is **two randomly-generated blocks**.

Q5.5 (2 points) What must be true of $M = (M_1, M_2)$ for this strategy to work?

Note: ? denotes a randomly-chosen value.

- ○ (A) $M_1 = 0$ and $M_2 = ?$
- ○ (B) $M_1 = ?$ and $M_2 = 0$
- ○ (C) $M_1 = ?$ and $M_2 = ?$
- ○ (D) $M_1 \neq M_2$
- ○ (E) $M_1 = M_2 \oplus 1$
- ○ (F) $M_1 = M_2$

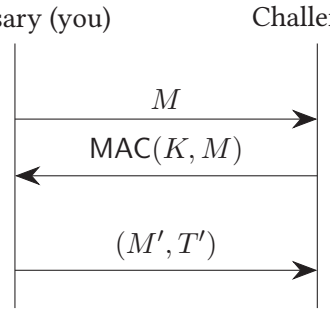
Q5.6 (4 points) Assume that M satisfies the condition you gave for the previous subpart.

Let $C = (IV, C_1, C_2)$ be the challenge ciphertext. Provide a strategy to guess whether M or M' was picked, with non-negligibly higher than 50% probability.

Your answer should be formatted along the lines of "If $C_1 \oplus 161 = 0$, then guess M, else guess M'" (no relation to actual solution).

Q6 *A Song of MACs and Signatures - Cryptography* **(16 points)**

EvanBot wants to review alternatives to HMACs and signatures. Below is a simplified version of the EU-CMA (referred to as EU-CPA in lecture) security game, with **only 1 query message** M (which will be sufficient for all subparts).



First, you send one query message M to the challenger.

Then, the challenger returns the MAC of M.

Finally, you output some $M' \neq M$ and a tag T', and win the game if T' is a valid MAC for M' with non-negligible probability.

In each subpart, select whether the given scheme is EU-CMA secure. If you selected "Insecure", provide an attack to win the EU-CMA game with non-negligible probability. If you selected "Secure", leave the boxes blank.

For all subparts: **if a box can be an arbitrary value, you must put "anything"** as the answer.

Q6.1 (4 points) $\mathsf{MAC}(K, M) = \mathsf{CBC}(K, H(M)) = (IV, C)$

CBC is AES-CBC encryption. IV is randomly generated per MAC. H has an output of 128 bits.

○ (A) Secure ○ (B) Insecure

Query Message:

Let $T = (IV, C)$ be the tag received for the query message M. Now provide a pair (M', T') such that $M' \neq M$ and T' is a valid MAC for M' with non-negligible probability.

Solution Message:

Note: Recall that the tag in this scheme is a pair of the form (IV, C).

Solution Tag:

Remember: **if a box can be an arbitrary value, you must put "anything"** as the answer.

Q6.2 (4 points) $\mathsf{MAC}(K, M) = \mathsf{CTR}(K, H(M)) = (IV, C)$

CTR is AES-CTR encryption. IV is randomly generated per MAC. H has an output of 128 bits.

○ (A) Secure ○ (B) Insecure

Query Message:

Let $T = (IV, C)$ be the tag received for the query message M. Now provide a pair (M', T') such that $M' \neq M$ and T' is a valid MAC for M' with non-negligible probability.

Solution Message:

Note: Recall that the tag in this scheme is a pair of the form (IV, C).

Solution Tag:

For each of the following signature schemes, answer whether the scheme is EU-CMA secure.

The EU-CMA game for signature schemes is identical to the game for MACs, except the challenger returns the signature of the query message under the secret key SK for their public key PK. Your goal as the adversary is to output a valid message/signature pair (M', S') for PK with M' different from the query message.

Q6.3 (4 points) $\mathsf{Sign}(SK, M) = M^d \bmod N$

$d = SK$ is an RSA private key associated with the public key (e, N).

M must satisfy $2 \leq M \leq N - 2$.

○ (A) Secure ○ (B) Insecure

Query Message:

Let S be the signature received for the query message M. Now provide a pair (M', S') such that $M' \neq M$ and S' is a valid signature for M' with non-negligible probability.

Solution Message:

Solution Signature:

Q6.4 (4 points) $\mathsf{Sign}(SK, M) = H(M) + xM \bmod p$

$x = SK$ is the private key chosen uniformly at random $\bmod p$, with the public key $PK = g^x$.

M must satisfy $2 \leq M \leq p - 2$.

$\mathsf{Verify}(PK, (S_1, S_2))$ returns `true` if $g^{-H(M)} \cdot g^S = (PK)^M \bmod p$.

Clarification after exam: $\mathsf{Verify}(PK, (S_1, S_2))$ *should read* $\mathsf{Verify}(PK, S)$.

○ (A) Secure ○ (B) Insecure

Query Message:

Let S be the signature received for the query message M. Now provide a pair (M', S') such that $M' \neq M$ and S' is a valid signature for M' with non-negligible probability.

Solution Message:

Solution Signature:

This page intentionally left (mostly) blank.

The exam continues on the next page.

Q7 *Be My Proxy? - Asymmetric Cryptography* **(18 points)**

Consider the following variant of ElGamal encryption. For all of Q7, assume that H outputs 128 bits.

Key Generation:

1. Choose a random private key $b \bmod p$ such that $\gcd(b, p-1) = 1$.
2. Derive the public key as $B = g^b \bmod p$.

Encryption:

1. Choose a random $r \bmod p$ such that $\gcd(r, p-1) = 1$.
2. Compute $R = g^r \bmod p$.
3. Let $K = H(B^r \bmod p)$ (i.e. the hash of $B^r \bmod p$).
4. Send $(C_1, C_2) = (R, \mathsf{Enc}(K, M))$.

Decryption:

1. Compute $K = H(\_\_\_\_\_)$.
2. Decrypt $M = \mathsf{Dec}(K, C_2)$.

Q7.1 (1 point) What goes in the blank in the decryption protocol?

○ (A) $C_1^b \bmod p$ ○ (B) $C_1^B \bmod p$ ○ (C) $B^{C_1} \bmod p$ ○ (D) $B^r \bmod p$

Q7.2 (3 points) Select all true statements.

☐ (A) The variant scheme is IND-CPA secure.

☐ (B) The variant scheme is multiplicatively malleable (e.g. a ciphertext C encrypting M can be transformed into a ciphertext C' encrypting $2M$, without knowing b).

☐ (C) The variant scheme is additively malleable (e.g. a ciphertext C encrypting M can be transformed into a ciphertext C' encrypting $M+1$, without knowing b).

☐ (D) None of the above

Q7.3 (2 points) Recall that the ElGamal scheme from lecture defines $C_2 = M \cdot B^r \bmod p$ instead of $\mathsf{Enc}(H(B^r \bmod p), M)$.

Alice and Bob believe that this variant scheme will protect them against a man-in-the-middle attack from Mallory, unlike lecture ElGamal. Assume that Alice and Bob do **not** know each other's public keys and must first share them over the insecure channel.

Is this correct?

○ (A) Yes, because Mallory can't predictably modify C_2.

○ (B) Yes, because $M \cdot B^r \bmod p$ is not confidential (i.e. it leaks some information about M).

○ (C) No, because Enc only provides authenticity if the attacker doesn't know the key.

○ (D) No, because Mallory can still cause Alice and Bob to derive keys known to Mallory.

Q7.4 (3 points) The hardness of which cryptographic problems is necessary for the variant scheme to be secure? Select all that apply.

- ☐ (A) Discrete logarithm problem
- ☐ (B) Diffie-Hellman problem
- ☐ (C) RSA problem
- ☐ (D) None of the above

Alice is about to leave on a month-long vacation, and wants the central mail server at her office to redirect all her email to Bob's inbox. However, since she uses encrypted email, Bob won't be able to read these messages as they were encrypted with B_{Alice} (Alice's public key).

They decide to use this ElGamal variant to develop a **proxy re-encryption** system. This system allows transforming ciphertexts encrypted with B_{Alice} to be encrypted with B_{Bob} instead, while keeping the underlying plaintext the same.

Q7.5 (6 points) Design a proxy re-encryption protocol using the modified ElGamal scheme. That is, design an algorithm to transform $C = (C_1, C_2) = (g^r \bmod p, \mathsf{Enc}(H(B_{\text{Alice}}^r \bmod p), M)$ encrypting some message M into $C' = (C_1', C_2')$ that decrypts to the same message M when decrypted by Bob with b_{Bob}.

Clarification after exam: *The original subpart had a typo, saying* $C_2 = \mathsf{Enc}(K, H(B_{Alice}^r \bmod p))$ *instead of the correct* $\mathsf{Enc}(H(B_{Alice}^r \bmod p), M)$ *as given in the protocol.*

First, the mail server is given a specific value V that will enable proxy re-encryption.

V:

- ○ (A) $b_{\text{Alice}} \cdot b_{\text{Bob}}^{-1} \bmod (p-1)$
- ○ (B) $b_{\text{Bob}} \cdot b_{\text{Alice}}^{-1} \bmod (p-1)$
- ○ (C) $b_{\text{Bob}} \cdot b_{\text{Alice}} \bmod (p-1)$
- ○ (D) $b_{\text{Bob}} + b_{\text{Alice}} \bmod (p-1)$

Given $C = (C_1, C_2)$ and V, give an expression for $C' = (C_1', C_2')$:

C_1':

- ○ (A) C_1
- ○ (B) $C_1 + V \bmod (p-1)$
- ○ (C) $C_1 \cdot V \bmod p$
- ○ (D) $C_1^V \bmod p$

C_2':

- ○ (A) C_2
- ○ (B) $C_2 + V \bmod (p-1)$
- ○ (C) $C_2 \cdot V \bmod p$
- ○ (D) $C_2^V \bmod p$

Q7.6 (3 points) Recall that the ElGamal scheme from lecture defines $C_2 = M \cdot B^r \bmod p$ instead of $\mathsf{Enc}(H(B^r \bmod p), M)$.

Is it still possible to create a proxy re-encryption scheme with lecture ElGamal?

- ○ (A) Yes, with an identical setup
- ○ (B) Yes, but with a modified setup
- ○ (C) No

Post-Exam Activity

EvanBot is having a post-midterm party! What did they cook?

Artwork by Shigezaki *Interested in having your art featured? Email evanbot@berkeley.edu.*

Comments/Assumptions Box

Congratulations for making it to the end of the exam! Feel free to leave any thoughts, comments, feedback, or doodles here. These comments won't affect your grade.

If there's anything else you want us to know, or you feel like there was an ambiguity in the exam, please put it in the box below. For ambiguities, you must qualify your answer and provide an answer for both interpretations. For example, "if the question is asking about A, then my answer is X, but if the question is asking about B, then my answer is Y". You will only receive credit if it is a genuine ambiguity and both of your answers are correct. We will only look at ambiguities if you request a regrade.