Paxson Spring 2013 — CS 161 Computer Security — Final Exam

PRINT your name: _____, _____
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct on this exam will lead to a "F"-grade for the course and that the misconduct will be reported to the Center for Student Conduct.

SIGN your name: _____

PRINT your class account login: `cs161-`_____ and SID: _____

Your TA's name: _____

Number of exam of person to your left: _____

Number of exam of person to your right: _____

You may consult two sheets of notes (each double-sided). You may not consult other notes, textbooks, etc. Calculators and computers are not permitted. Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there.

You have 170 minutes. There are 9 questions, of varying credit (300 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	37	30	32	38	29	38	34	30	32	300
Score:										

Problem 1 ***Impersonation*** **(37 points)**

(a) (12 points) For each of the following, circle either **TRUE** or **FALSE**.

As a web site operator, one way to fully defend your users against clickjacking is to avoid using frames on your site. **TRUE** **FALSE**

Sandboxing brower components is an effective technique for defending users against phishing. **TRUE** **FALSE**

Websites use frame-busting to ensure that their pages aren't included in a frame on another site. **TRUE** **FALSE**

Most browsers allow Javascript to modify the appearance and offset of a cursor. **TRUE** **FALSE**

Phishing can only occur if a victim clicks on a link of the phisher's devising. **TRUE** **FALSE**

Spear-phishing has a higher rate of success than normal phishing. **TRUE** **FALSE**

(b) (25 points) Briefly answer each of the following:

1. Identify an accessibility issue with the use of CAPTCHAs.

   Solution: One issue is blind users (potentially addressed by audio CAPTCHAs). Another is simply the difficulty of reading today's CAPTCHAs even for those with no visual impairment.

2. Do today's CAPTCHAs effectively prevent modern attackers from registering large numbers of accounts (each of which requires solving a CAPTCHA to complete the registration)?

   Circle **YES** or **NO** and briefly explain why.

   Solution: NO: by purchasing CAPTCHA solving services (often done by "outsourcing", that is, paying people to solve CAPTCHAs), attackers can overcome the impediments to large-scale account registration presented by CAPTCHAs.

   Solutions of **YES** accompanied by a discussion of limited attacker resources are also fine.

3. Suppose a site implements a CAPTCHA by presenting users with four images and asking them to identify the one that shows a dog. Discuss a weakness with this particular approach (different from any criticism of CAPTCHAs you discussed in the earlier questions).

   Solution: One weakness: if the challenge is just to identify one correct image among four, an attacker can simply guess and if they fail, try again until they succeed.

   Or: if the images come from some sort of library, the attacker can potentially build up a dictionary of the known images.

   Or: attackers can employ algorithms to detect different types of animals. For example, they might be able to script up use Google's "image search" capability to automatically determine the correct answer.

   Or: if such images are distorted like has been the case for textual CAPTCHAs, it might be difficult for legitimate users to determine which image is correct.

4. Does ensuring *visual integrity* sufficiently protect against clickjacking?

   Circle **YES** or **NO** and briefly explain why.

Solution: NO: there are two basic approaches exploited by clickjacking, undermining visual integrity and undermining *temporal integrity*. The latter refers to tricks like changing what's beneath the cursor just as the user issues a click.

5. Briefly explain how *Browser-in-Browser* attacks can help phishers with conducting phishing attacks.

Solution: Browser-in-browser attacks construct images that appear to be separate browser windows but in fact are just pixels drawn within an existing browser window. Such images allow a phisher to fool a user into thinking that a window the user interacts with has all of the expected "security indicators" in the right place, such as padlocks or highlighting of URLs.

Problem 2 ***Intrusion Detection*** **(30 points)**

As a fresh IT security professional, your boss tasks you with evaluating an upcoming purchasing decision regarding several competing NIDS products. The three NIDS under consideration are *(i) AnomSpotter*, an anomaly-based zero-day detector, *(ii) JohnHancock*, a signature-based system, and *(iii) Spec-n-Behave*, a system that includes both specification-based and behavioral detection.

(a) For each of the following detection styles, briefly write down one advantage and one disadvantage:

- Anomaly-based
  - \+ *Can detect novel attacks*
  - – *Can miss known attacks. Often exhibits high false positive and/or false negative rates. Can be mis-trained on historical data that includes attacks.*
- Signature-based
  - \+ *Simple to understand. Easy to share.*
  - – *Can't detect novel attacks, or even variants. Depending on approach, can exhibit high false positives.*
- Specification-based
  - \+ *Can detect novel attacks. Can have low false positive rate.*
  - – *Lots of work to develop. Specifications often aren't shareable.*

(b) JohnHancock and AnomSpotter come with explicit guarantees on the rate of false positives, advertising $\frac{1}{100{,}000}$ and $\frac{1}{10{,}000}$ respectively. Spec-n-Behave comes with an explicit guarantee on the rate of false negatives, advertising $\frac{3}{1{,}000{,}000}$. **Circle** which tool would perform **best** in your company's environment:

- AnomSpotter
- JohnHancock
- Spec-n-Behave
- Impossible to determine with information given

Solution: Without knowing the base rate of attacks, the false negative rates for JohnHancock and AnomSpotter, and the false positive rate for Spec-n-Behave (also FYI as well as the "unit of analysis", i.e., the denominator for false positive/negative rates—not an issue we developed in class), it is impossible to make that decision.

(c) In order to understand the nature of threats your company faces, you dig through the incident reports of the past two years. In 2011, your company got 0wned through numerous well-known and unsophisticated remote code-exec exploits of unpatched services.

Circle which tool would have been **single most practical one** for **effectively** and **efficiently** detecting these attacks?

- AnomSpotter
- JohnHancock
- Spec-n-Behave
- *All would work equally well for this*

Solution: JohnHancock, because signatures often have enough power to detect well-known unsophisticated attacks, and are efficient at doing so, not requiring much work to set up.

(d) Looking then through the incident reports of 2012, you observe a shift towards compromises due to web attacks. The attacker tools operated in a very stealthy fashion, but a sharp member of the security staff found it was possible to identify the malicious web requests due the order of the HTTP headers in those requests. While HTTP does not require any particular order, the staffer noticed that benign requests happen to use a different order.

Assume you would not have known in advance that the attack traffic would have this appearance. **Circle** which tool would have been **single most practical one** for **effectively** and **efficiently** detecting these attacks?

- AnomSpotter
- JohnHancock
- Spec-n-Behave
- *All would work equally well for this*

Solution: AnomSpotter, because the detection mechanism relies on identifying an abnormal order of HTTP headers that normal clients don't exhibit. Spec-n-Behave would have to develop a specification requiring that headers appear in a particular order, but the problem states that HTTP does not itself impose any required order.

(e) **Circle** which system would require the **least** amount of initial effort for the security team to deploy?

- AnomSpotter

- JohnHancock
- Spec-n-Behave
- *All require about the same initial effort*

Solution: JohnHancock will be easiest to initially employ, since signature-based systems can come with libraries of signatures developed by others. AnomSpotter, on the other hand, needs a training period to establish a profile of normal activity, and Spec-n-Behave requires development of specifications, a time-consuming process.

Problem 3 ***Anonymity and Censorship*** **(32 points)**

(a) For each of the following, circle either **TRUE** or **FALSE**.

Paid 3rd party proxy services provide the same anonymity properties as Onion Routing networks. **TRUE** **FALSE**

Attackers can easily achieve strong anonymity while it is more difficult for the average Internet user (without the usage of networks such as Tor). **TRUE** **FALSE**

Censorship systems always work by identifying keywords. **TRUE** **FALSE**

Censorship systems and NIDS have similar properties. **TRUE** **FALSE**

(b) For each of the following, fill in the blank.

Assume correct usage of an Onion Routing network (such as Tor), configured to use 3 mixes. Of these 3, how many must be honest to preserve the confidentiality of your IP address?

Solution: 1 honest mix suffices.

State a generally effective way to defeat keyword-based censorship that doesn't rely upon anonymity:

Solution: Encryption will prevent the censor from recognizing the sensitive keywords.

For each of the following questions write a *short* explanation.

(c) **In at most 2 sentences**, describe an issue/attack that users of anonymity systems may encounter. Such an issue or attack must harm the anonymity of the user.

Solution: The identify of users can be revealed by their application-layer activity (for example, the social network account they log into). A user's DNS lookups, if not sent through the anonymity system, will enable tying their system to their subsequent activity. "Confirmation attacks" occur when an adversary can correlate activity from the user's system when it enters into the anonymity system versus the activity that exits the system.

(d) **In 1 sentence**, what can an in-path censor do that an on-path censor cannot?

Solution: In-path censors can directly drop or modify traffic, rather than having to rely upon injection to disrupt traffic.

(e) **In 1 sentence**, what advantage does an on-path censor have over an in-path censor?

Solution: On-path censorship has lower performance requirements (since the censorship device doesn't have to forward traffic), allowing greater scalability. Also, the failure of an on-path censorship device does not disrupt use of the network.

(f) You suspect you are being attacked by an on-path censorship device. **In at most 2 sentences**, describe how you can detect such an attack. Your technique should be able to differentiate between network connectivity problems and censorship.

Solution: You could attempt to fetch the same content from a different location and see whether the results differ. Or you could look for the traffic injected by the on-path censorship device, which will appear *in addition* to the regular traffic, rather than instead of it.

Problem 4 ***TLS and DNSSEC*** **(38 points)**

(a) Oski wants to securely communicate with CalBears.com using TLS. Which of the following things must Oski trust in order to communicate with confidentiality, integrity, and authenticity? **Circle all that apply**.

- [x] The operators of CalBears.com
- [x] Oski's Computer (HW and SW)
- [x] The designers of the cryptographic algorithms
- [ ] Computers on Oski's local network
- [ ] The operators of CalBears.com's authoritative DNS servers
- [ ] The entire network between Oski and CalBears.com
- [x] CalBears.com's Certificate Authority (CA)
- [x] All the CA's that come configured into Oski's browser
- [ ] All the CA's that come configured into CalBears.com's software
- [ ] The operators of .com's Authoritative DNS servers
- [ ] The operators of the Authoritative DNS Root servers

(b) For each of the following, circle either **TRUE** or **FALSE**.

TLS uses asymmetric cryptography to directly encrypt the contents of communications: **TRUE** **FALSE**

Certificates are a necessary component of TLS: **TRUE** **FALSE**

TLS always provides Perfect Forward Secrecy: **TRUE** **FALSE**

TLS always provides CalBears.com with authenticity of Oski's identity: **TRUE** **FALSE**

TLS allows both parties of a communication to contribute to key generation: **TRUE** **FALSE**

TLS allows the use of stream ciphers: **TRUE** **FALSE**

For transmitting data from a client to a server, a TLS session uses more than one key: **TRUE** **FALSE**

Clients finalize the choice of the TLS ciphersuite: **TRUE** **FALSE**

After taking CS161 you've seen the light and wish to help spread the gospel of DNSSEC. You realize one way to help speed up adoption would be combine DNSSEC and TLS in some way to eliminate a (perceived?) problem with TLS. *Hint: It's about trust.*

(c) Fill in the blanks: While TLS provides **channel** security, DNSSEC provides **object** security.

(d) Consider our goal of eliminating a *trust* problem with the existing TLS design, along with your answer to the previous question (4 c). Identify a key component of TLS that could instead rely on the DNSSEC infrastructure. It's ok if this modification requires changes to the usage of TLS and DNSSEC. **Only identify a single component**.

Solution: Certificates (or, more generally, "authentication").

(e) **In at most 2 sentences**, describe the change.

Solution: The basic idea would be for a TLS client to retrieve a site's public keys via DNSSEC records from the site's domain, rather than via a certificate sent by the server and signed by a CA. Such an approach could also instead return signatures of public keys that the server would then still send to the TLS client; the client would now validate the public key based on the signature received via DNSSEC rather than from some CA. The inspiration for this question came from DNS-based Authentication of Named Entities (DANE) https://en.wikipedia.org/wiki/DNS-based_Authentication_of_Named_Entities. DANE is a standard current under development that, among other things, allows certificates to be bound to DNSSEC records.

(f) Assume end-to-end DNSSEC deployment as well as full deployment of **your change** (*good job!*). Oski wants to securely communicate with CalBears.com using TLS. Which of the following things must Oski trust in order to communicate with confidentiality, integrity, and authenticity? **Circle all that apply**.

- The operators of CalBears.com
- Oski's Computer (HW and SW)
- The designers of the cryptographic algorithms
- Computers on Oski's local network
- The operators of CalBears.com's authoritative DNS servers
- The entire network between Oski and CalBears.com
- CalBears.com's Certificate Authority (CA)
- All the CA's that come configured

into Oski's browser

- All the CA's that come configured into CalBears.com's software

- The operators of .com's Authoritative DNS servers

- The operators of the Authoritative DNS Root servers

(g) **In at most 2 sentences**, why you think this change in trust is either good or bad?

Solution: Many answers are possible here. One could say it's a good change because there are now fewer parties to trust (the operators of the root name servers, the `.com` name servers, and CalBears.com itself). Another solution is that it's a good change because it associates the trust directly with the parties associated with a domain, rather than with all CA's. But one could also argue that now the operators of the root name servers gain a great deal of power.

Problem 5 ***Reasoning About Memory Safety*** **(29 points)**

The following code takes two strings as arguments and returns a pointer to a new string that represents their concatenation:

```
1  char *concat(char s1[], int n1, char s2[], int n2)
2  {
3    int i, j;
4    int n = n1 + n2;
5    char *s;
6
7    if (n1 < 0 || n2 < 0 || n <= 0) return 0;
8
9    s = malloc(n);
10   if (!s) return NULL;
11
12   for (i=0; s1[i] != '\0'; ++i)
13     s[i] = s1[i];
14
15   for (j=0; s2[j] != '\0'; ++j)
16     s[i+j] = s2[j];
17
18   s[i+j] = '\0';
19   return s;
20 }
```

The function is intended to take two strings and return a new string representing their concatenation. If a problem occurs, the function's expected behavior is undefined.

(a) For the three statements assigning array elements, write down `Requires` predicates that must hold to make the assignments memory-safe:

1.
```
/* "Requires" for line 13:
 *   s1 != NULL && s != NULL && i >= 0 &&
 *    i < size(s1) && i < size(s)  [same as i < n]
 */
s[i] = s1[i];
```

2.
```
/* "Requires" for line 16:
 *   s2 != NULL and s != NULL && j >= 0 && i+j >= 0 &&
 *    j < size(s2) && i+j < size(s)
 */
s[i+j] = s2[j];
```

3.
```
/* "Requires" for line 18:
 *   s != NULL && i+j >= 0 && i+j < size(s)
 *
 */
s[i+j] = '\0';
```

(b) Here is the same code again, with more space between the lines. Indicate changes (new statements or alterations to the existing code, plus a relevant precondition for calling the function) necessary to ensure memory safety. Do *not* change the types of any of the variables or arguments.

```
/* Precondition:
 *
 * s1 != NULL && s2 != NULL &&
 * n1 <= size(s1) && n2 <= size(s2)
 *
 */
1  char *concat(char s1[], int n1, char s2[], int n2)
2  {
3    int i, j;
4    int n = n1 + n2 + 1; /* FIX FENCEPOST ERROR */
5    char *s;
6
7    if (n1 < 0 || n2 < 0 || n <= 0) return 0;
8
9    s = malloc(n);
10   if (!s) return NULL;
11
12   for (i=0; i < n1 && s1[i] != '\0'; ++i) /* CHECK BOUNDS */
13     s[i] = s1[i];
14
15   for (j=0; j < n2 && s2[j] != '\0'; ++j) /* CHECK BOUNDS */
16     s[i+j] = s2[j];
17
18   s[i+j] = '\0';
19   return s;
20 }
```

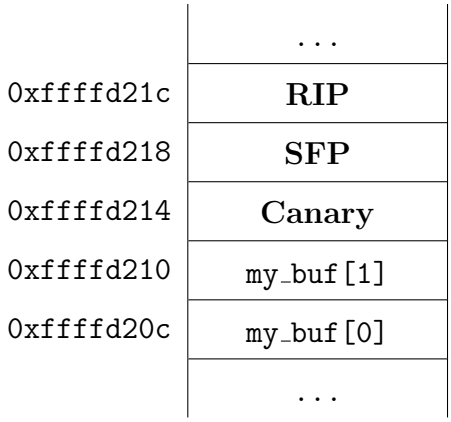
Problem 6 *Buffer Overflow Defenses and Exploits* (38 points)

(a) The following code is compiled using an **IA-32 (x86)** compiler that implements *stack canaries*:

```
void copy()
{
    int my_buf[2];
    ... /* Point where we draw the stack diagram */
}
```

In the following stack diagram, write in each empty box the value that would be held there once we reach the point specified in the `copy()` code (assume we've given you the correct region of the stack). Use the following values:

Canary — `my_buf[0]` — `my_buf[1]` — RIP — SFP



(b) The following table lists four defenses against buffer overflows. For each defense, **mark with an X** which of the following attack methods the defense will **prevent**. The attacker can only place shellcode in buffers **on the stack**. Consider techniques **in isolation**, not combined. If the defense fails to prevent all of these methods, mark the entry for "*None*".

A *Return-oriented programming (ROP)*
B Forms of *Arc injection* besides ROP
C Return to stack pointer
D Function pointer overwrite
E Saved frame pointer overwrite
F Local control variable overwrite

	A	**B**	**C**	**D**	**E**	**F**	*None*
Stack canaries	**X**	**X**	**X**		**X**		
Bounds checking	**X**	**X**	**X**	**X**	**X**	**X**	
Stack address randomization				**X**	**X**		
Non-executable stack			**X**	**X**	**X**		

Solution:

Because we consider techniques in isolation, you can't combine any techniques with Arc injection. For A and B, you need to be able to write known non-stack addresses to the RIP. For C (Project 1 Question 5), you need to be able to overwrite the RIP to execute code on the stack. For D and E (Project 1 Question 4), you need to be able to write known addresses to a local function pointer/the SFP to execute code on the stack. For F, you need to be able to overwrite a local variable on the stack.

A defense will prevent an attack method if an attacker can no longer with high probability execute malicious (sequences of) code.

Stack canaries: You can't overwrite anything above the Canary, so neither RIP and SFP. Prevents: A,B,C,E

Bounds checking: You can't overwrite anything outside the buffer. Prevents: All.

Stack address randomization: You can overwrite the entire stack, but you don't know the address of shellcode placed on the stack. Prevents: D,E

Non-executable stack: You can't execute code on the stack. Prevents: C,D,E

Finally, here is an example of code vulnerable to F:

```
1 void fn(char* buf, unsigned int n)
2 {
3     int authenticated=0;
4     char my_buf[16];
5     memcpy(my_buf,buf,n);
6     if (authenticated)
7     {
8         //do something
9     }
10 }
```

Problem 7 ***Game of P0wns*** **(34 points)**

(a) For each of the following, circle **True** if the statement is correct, or **False** if it is not.

—— **False** To date, the Stuxnet worm is the fastest-spreading worm that has appeared in the Internet.

—— **False** Worms that spread by randomly scanning Internet addresses will grow exponentially fast all the way until they have infected the entire vulnerable population.

—— **False** A common approach for creating metamorphic viruses uses encryption technology.

—— **False** Eliminating buffer overflows would completely prevent the problem of Internet worms.

—— **False** Eliminating buffer overflows would completely prevent the problem of botnets.

True —— Some modern intrusion detection systems analyze the effects of executing web programs (such as Flash) in controlled environments.

True —— *Social engineering* refers to fooling users into taking insecure actions.

True —— Malware has been known to sometimes cause physical hardware damage to infected systems.

—— **False** Internet worms and viruses first became a problem with the arrival of Code Red in 2001.

True —— Some viruses add their code to that of existing executables residing on disk.

—— **False** *Botnet Command-and-Control* refers to the US government's command center that tracks malware epidemics as they propagate.

—— **False** The largest Internet worm ever infected approximately 400,000 systems.

—— **False** The longest that an Internet worm outbreak has persisted (systems remained infected by the worm) is approximately 3 months.

True —— One technique used by some Internet worms has been to inspect files on a newly infected system to find other systems to try to infect.

—— **False** Modern malware infections typically use multiple "zero day" exploits.

—— **False** *Bullet-proof hosting* refers to use of hardened web sites that provably have no XSS or SQL vulnerabilities.

——— **False** In the context of botnets, *externalities* refers to the ability for botmasters to host their servers in countries outside the legal reach of the United States.

True ——— *Transaction generators* are a type of malware that infects a user's browser and issues illicit banking transactions after the user has authenticated to their bank's web site.

——— **False** *Click fraud* refers to a type of phishing where users believe they clicked on a link taking them to a trusted site like a bank, but instead they go to the phisher's site.

True ——— Users who attempt to buy products advertised by email spam usually receive a functional product in return.

True ——— In the context of the underground economy, *affiliates* are business partners who receive commissions (payment) for providing some service, such as bringing users to an underground site who then themselves make purchases.

——— **False** The term *money mules* refers to hardworking members of the underground economy who will work many different aspects of an Internet attack in order to ultimately gain a profit.

True ——— *Pump-and-dump* refers to monetizing email spam by inducing fluctuations in the prices of certain stocks.

True ——— The underground economy service of *Pay-Per-Install* is one of the reasons why when systems become infected today, they often wind up having multiple infections at the same time.

(b) List 4 goods and/or services that attackers can buy on the "underground economy". Each should be quite different from the others.

Solution: There are many possibilities: for example, stolen accounts, cashout, "confirmers", DDoS-for-hire, exploit kits, stolen credit cards, synthetic accounts.

(c) Roughly how much does it cost on the underground to purchase installations of malware on multiple Internet systems? State a quantity, a price, and one factor that influences the price.

Solution: The pricing presented in lecture ranged from about $6 for 1,000 installs (systems in LB = Lebanon, presumably) to $150 for 1,000 installs (US systems). Anything in this basic range sufficed for credit regarding quantity-and-price.

A major factor that influences price is geographic region. It's also reasonable

to simply frame that price responds to supply-and-demand, given the business nature of how pay-per-install services operate.

Problem 8 ***Bugs in Key Generation*** **(30 points)**

This problem concerns the impact of potential bugs in cryptographic key generation. The threats are Eve, an eavesdropper, and/or Mallory, a MITM attacker. Assume neither Eve nor Mallory can conduct massive brute-forcing attacks ($\geq 2^{64}$ attempts).

(a) In 2011, a bug was found in the library provided with Ruby for generating RSA public/private key pairs. The buggy library would always generate $e = 1$. The library would then compute the corresponding d using the normal algorithm for deriving d from e.

For this problem, assume that Alice wants to **send a *single* message** to Bob (one-way communication), solely using public-key cryptography for security. Assume that Alice and Bob have a secure means for exchanging their public keys.

1. Suppose Alice generated her public/private key pair using the buggy Ruby library, but Bob used a secure library without the flaw. If Alice communicates with Bob using RSA-based cryptography for **confidentiality**, **Circle** which of the following describes the impact of the bug:

   - **No effect in this scenario:** For Alice to send a message confidentially to Bob, she encrypts it with Bob's public key, not with her own.
   - Bob can read Alice's message, but so can Eve
   - Bob will not be able to read Alice's message

2. Suppose instead Bob generated his public/private key pair using the buggy Ruby library, but Alice used a secure library without the flaw. Again, if Alice communicates with Bob using RSA-based cryptography for **confidentiality**, **Circle** which of the following describes the impact of the bug:

   - No effect in this scenario
   - **Bob can read Alice's message, but so can Eve:** Bob's use of $e = 1$ means that when Alice encrypts her message M, she sends to Bob $M^e \mod N = M^1 \mod N = M$ (given $M < N$), so she in fact transmits M in the clear. This enables Eve to read it (as well as Bob).
   - Bob will not be able to read Alice's message

3. Suppose Alice generated her public/private key pair using the buggy Ruby library, but Bob used a secure library without the flaw. If Alice and Bob use RSA-based public key cryptography for **authentication** of a message **sent by Alice to Bob** (do not concern yourself with messages from Bob to Alice), **Circle** which of the following describes the impact of the bug:

   - No effect in this scenario
   - Alice's message will not verify to Bob as properly signed
   - **Mallory can construct a message that appears to have a valid signature from Alice:** To sign message M, Alice transmits $M^d \mod N$.

For $e = 1$, its inverse is $d = 1$, so Alice again sends M unmodified (given $M < N$). When Bob verifies the signature using $M^{de} = M^{11} = M$ it indeed verifies correctly. However, Mallory can also send a message M' to Bob with signature M' and it will likewise verify corretly to Bob.

4. Suppose instead Bob generated his public/private key pair using the buggy Ruby library, but Alice used a secure library without the flaw. Again, if Alice and Bob use RSA-based public key cryptography for **authentication** of a message sent by Alice to Bob, **Circle** which of the following describes the impact of the bug:

   - **No effect in this scenario:** Because the authentication of Alice's messages are based on Alice's RSA public/private key pair, and not Bob's, that Bob has generated a flawed pair will not matter.
   - Alice's message will not verify to Bob as properly signed
   - Mallory can construct a message that appears to have a valid signature from Alice

(b) Instead of the Ruby library bug occurring when generating RSA private/public key pairs, suppose the library had a bug where for Diffie-Hellman key exchanges it always used $g = 1$.

If both Alice and Bob used the library with this new bug, **Circle** which of the following describes the impact of the bug:

- No effect in this scenario
- Alice and Bob will fail to agree upon the same key using the key exchange
- **Alice and Bob will agree upon the same key, but Eve can also determine the value of the key:** For the D-H exchange, Alice will send to Bob $g^a = 1^a = 1$. Bob will likewise send to Alice $g^b = 1^b = 1$. When Alice computes $g^{ba} = 1^a = 1$, she gets the same key value as Bob will. However, Eve will also know that this key is equal to 1.
- Alice and Bob will agree upon the same key, and while Eve cannot determine the value of the key, Alice and Bob lose *perfect forward secrecy*.

(c) Suppose Alice and Bob send new messages to each other every few minutes, so they decide to establish long-term **confidential** communication using symmetric-key cryptography. To do so, they first securely agree upon a secret key, K. They then use a library that employs *AES-256* in *CTR* mode as a stream cipher. The CTR mode counter is managed in the usual fashion. The library generates a fresh nonce every hour.

Circle which of the following best describes this scenario:

- Alice and Bob will be able to communicate confidentially.

- **In some situations, Eve may be able to learn some information about the contents of Alice and Bob's communication.:** Proper use of a stream cipher includes changing the nonce for every new message. Since their library instead only changes it every hour, and they send multiple messages per hour, this means that the same nonce will be used for multiple messages. That will result in the same pseudo-random bitstream for the messages, so by XOR'ing ciphertext together, Eve can recover relationships between the plaintext bits. This will not, however, enable Eve to learn anything about the key used to generate the bitstream, and Eve can't *consistently* recover complete messages because doing so requires known plaintext for some of the messeages.
- In some situations, Eve may be able to learn some information about the contents of Alice and Bob's communication *and* some information about K.
- Eve can consistently recover complete messages from Alice and Bob's communication.
- Eve can consistently recover complete messages from Alice and Bob's communication *and* can recover K.

Problem 9 ***Composition of Cryptographic Tools*** **(32 points)**

For this problem, assume that Alice wants to send a *single* message M to Bob. To do so, Alice and Bob can potentially use a number of different approaches and cryptographic technologies, which we will describe using the following terminology:

M	Plaintext for a single message
$A \mid\mid B$	Concatenation of A with B. Assume the receipient can unambiguously decompose this back into the original values of A and B.
K_A	Alice's public key
K_A^{-1}	Alice's corresponding private key
K_B	Bob's public key
K_B^{-1}	Bob's corresponding private key
E_K	Public-key encryption using RSA with the public key K
$\text{Sign}_{K^{-1}}$	Public-key signing using RSA with the private half of K.
s_k	Symmetric cryptography key
AES_{s_k}	Symmetric-key encryption using AES-256 in CBC mode, with the key s_k
AES-EMAC_{s_k}	Keyed MAC function presented in lecture, using the key s_k
PRNG_{s_k}	Bit-stream from a cryptographically strong pseudo-random number generator, seeded with s_k
IV	An Initialization Vector randomly generated for each use
SHA	SHA-256 hash function

You can assume that the public keys have been securely distributed, so Alice and Bob know their correct values.

Consider the following properties that Alice and Bob might desire their communication to have: *Confidentiality*, *Integrity*, *Authentication*, and *Non-Repudiation*. For each of the following possible communication approaches, **circle** which of these properties will securely hold in the presence of Mallory, a MITM attacker. Circle **None** if *none* of the properties hold. If an approach fails entirely (requires a step that either Alice or Bob cannot execute), circle **Broken** (and do not bother with the others).

(a)

(b) Alice sends to Bob: $E_{K_A}(M \mid\mid \text{Sign}_{K_A^{-1}}(\text{SHA}(M)))$

> **Solution: Broken**—to decrypt with this scheme, Bob needs to possess Alice's private key.

(c) Alice sends to Bob: $E_{K_B}(M \mid\mid \text{Sign}_{K_B^{-1}}(\text{SHA}(M)))$

Solution: Broken—this scheme requires Alice to possess Bob's private key for the signing operation.

(d) Alice sends to Bob: $E_{K_A}(M)$, $\text{Sign}_{K_B^{-1}}(\text{SHA}(M))$

Solution: Broken—to decrypt with this scheme, Bob needs to possess Alice's private key. Alice also needs to possess Bob's private key for the signing operation.

(e) Alice sends to Bob: $E_{K_B}(M)$, $\text{Sign}_{K_A^{-1}}(\text{SHA}(M))$

Solution: Provides all of *Confidentiality* (via the encryption using Bob's public key), *Integrity* (via the digital signature over the hash of the message), *Authentication* (likewise) and *Non-Repudiation* (via Alice using her private key for the digital signature).

It's valid to note that Eve can exploit this structure to conduct a *confirmation* attack, because the using of signing allows Eve to determine whether M had a given value. That means the approach would no longer have full *Confidentiality*.

(f) Alice generates a new symmetric key s_k and sends to Bob:
$E_{K_A}(s_k)$, $E_{K_B}(s_k)$, $\text{AES}_{s_k}(M)$

Solution: Only provides *Confidentiality*. While Bob cannot recover s_k from $E_{K_A}(s_k)$ (because Bob lacks Alice's private key), he can do so from $E_{K_B}(s_k)$. By itself, AES does not provide integrity or authentication, so this scheme only provides *Confidentiality*, and because Alice does not sign her message, it also lacks non-repudiation.

(g) Alice generates a new symmetric key s_k and sends to Bob:
IV, $E_{K_A}(s_k)$, $E_{K_B}(s_k)$, $M \oplus \text{PRNG}_{s_k} \oplus IV$

Solution: This scheme is similar to the previous one, and only provides *Confidentiality*. The encryption using $M \oplus \text{PRNG}_{s_k}$ is the same as a stream cipher, which provides confidentiality, but without a MAC, the communication lacks integrity. The addition of an Initialization Vector does not gain integrity or

authentication (it helps with providing confidentiality for *multiple* messages, which the problem framing specifically did not ask for).

(h) Alice generates new symmetric keys s_{k_1} and s_{k_2}, and sends to Bob: $E_{K_A}(s_{k_1})$, $E_{K_A}(s_{k_2})$, $E_{K_B}(s_{k_1})$, $E_{K_B}(s_{k_2})$, $\text{AES}_{s_{k_1}}(M)$, $\text{AES-EMAC}_{s_{k_2}}(\text{SHA}(M))$, $\text{Sign}_{K_A^{-1}}(\text{SHA}(s_{k_1}))$, $\text{Sign}_{K_A^{-1}}(\text{SHA}(s_{k_2}))$

Solution: This scheme provides *Confidentiality* (via the use of AES) and *Integrity* (via use of the keyed MAC function), and hence *Authentication*. It does not provide non-repudiation because the integrity/authentication of M does not demonstrate possession of Alice's private key.

Here again one could argue that it does not fully achieve *Confidentiality* because the MAC could conceivably leak a bit of information about $\text{SHA}(M)$, which in turn allows a bit of headway on confirmation attacks.

(i) Alice and Bob privately exchange symmetric keys s_k and s'_k in advance. Alice later uses these keys to send to Bob: IV, $\text{AES}_{s_k \oplus IV}(M)$, $\text{AES-EMAC}_{s'_k \oplus IV}(\text{SHA}(M))$

Solution: This scheme provides *Confidentiality* because of the use of AES with a key that the attacker cannot guess ($s_k \oplus IV$), and *Integrity* and *Authentication* due to the use of a (different) unguessable key for the MAC.