CS 166 MIDTERM 1 SOLUTION

Overview

Marks on question

- E: easy/straightforward - either appeared in the Aor on the lecture notes;
- M: medium - not explicitly on notes, but explained in class, or require some understanding;
- H: hard - didn't show in class but should be able to answer if understand the material.

General rules of grading for short answers
(see comments under each question on your exam for deduction reasons):

- grading is mainly based on your reasoning/explanation, not the conclusion you got
- -0.5 for minor errors
- -1 (or 0.75, depends on whether the question has -0.5 rank) for major errors (but at least makes sense)
- -1.5 for anything completely wrong
- You get 0.5 for each short answer question (not each sub-question though) as long as you wrote something meaningful (except for AI-generated/copy from somewhere – you get 0 on the question) .

Matching & Multiple Choices

E - Q1: Basic terminologies - how to speak crypto (L1 P8)

(You were asked 2 out of the following 3)

Making “secret codes” --> **Cryptography**
Breaking “secret codes” --> **Cryptanalysis**
Making and breaking “secret codes” --> **Cryptology**

E - Q2: Think like Trudy: different software-based attacks (L9 P18 - 19 & L12 P5 & A3 Q7)

(You were asked 2 out of the following 3)

You found that a web service only checks the input on the client side but NOT on the server side
--> **Incomplete mediation attack**
You found there's some delay between withdrawing money from ATM and the remaining balance being updated.
--> **Race condition attack**
You are a programmer of a bank to implement transactions between accounts.
--> **Salami attack**

E - Q3: CIA and non-repudiation (L4 P12 & L7 P9 & A2 Q10)

(You were given different options, here are the options that are correct)

Encryption provides confidentiality
MAC provides integrity
Signature by public key crypto provides integrity and non-repudiation

Yan Chen
Computer Science Department

E - Q4: Malware timeline (L10 P7 - 10 & A3 Q1)

(You were given different options, here are the options that are correct)

In 1980s, malware didn't spread very fast because the Internet was not that advanced yet.
In 2000s, malware started to spread faster and faster but the attackers were just showing off their ability without doing something too harmful.
Recently (~2010s), attackers started to attack other's computers for profit.

E - Q5: Public key crypto misinterpretations (L5 P5 - 6 & L7 P10 - 11 & A2 Q1 Q2 Q8)

($C = \{[M]_{Alice}\}_{Bob}$; and you were given different options, here are the options that are correct)

To read M, Bob needs his private key and Alice's public key.
To compute C, you need Bob's public key and Alice's private key.

E - Q6: Block cipher CBC mode decryption auto recover (L4 P9 - 10 & A1 Q15)

(You were given different n's from n = 0 to n = 3)

Given $P_n = C_{n-1} \oplus D(C_n, K)$, changing C_n will result in wrong P_n and P_{n+1} for n = 0 to 3

E - Q7: Evade signature detection (L10 P15 - 17 & A3 Q3)

(You were given different options, here are the options that are correct)

When encrypting the malware, there's no need for attackers to choose a strong cipher.
Polymorphic worm means the decryptor is mutated so there's no common signature for decryptors.
We can run polymorphic worm in a sandbox (emulator) and then search for malware's signature.
Metamorphic worm means the worm is mutated so there's no common signature.
Detecting metamorphic worm is hard.

E - Q8: Crypto in general (L1 P20 & L4 P6 & L8 P6 & P21 & A1 Q3 Q13 & A2 Q12 Q13)

(You were given different options, here are the options that are correct)

By the definition of "secure" we covered in class, a secure cipher may be easier to break compared to an insecure cipher.
By the definition of "secure" we covered in class, an insecure cipher may be harder to break compared to a secure cipher.

We can use any mode to encrypt multiple blocks; it doesn't matter how each block is encrypted.

There will be collisions even using a secure crypto hash function.
One usage of hash we covered in class is to reduce the amount of spam emails.

E - Q9: Malware detection (L10 P11 - 14 & A3 Q2)

(You were given different options, here are the options that are correct)

Signature detection can be used alone.
Signature detection can tell which software/app is malware.
Signature detection may trigger false alarms.

Change detection can detect infection from an unknown malware.
Change detection has zero false negative rate in theory.
Change detection has high false alarm rate.

Anomaly detection is trying to detect "big" behavior/usage changes.
Anomaly detection can detect infection from an unknown malware.
Anomaly detection may have high false alarm if "normal" is not updated.

E - Q10: Functions of AES (L3 P18 - 21 & A1 Q10)

Only ShiftRow is for diffusion, others all for confusion.

E - Q11: Cube root attack on RSA when e = 3 (L6 P12 & A2 Q6)

(You were given different N's, and you need to choose correct M's)

The M such that $M^3 < N$ are those would cause cube root attack since it means $C = M^3 \bmod N = M^3$

Fill-in-blanks

E - Q12: Find the key for a parameterized Caesar cipher (L1 P15 & A1 Q1)

(You were given different ciphertext) Key should be the **number** of shifts.
A quick way is to count based on the mapping from plaintext 'a' to the corresponding ciphertext.

E - Q13: Using S-Box in DES (L3 P13 & A1 Q9)

(You were given different input) As given in the hint, bits 0 and 5 for row; bits 1 through 4 for column.
Just find the "intersection" of row and column.

E - Q14: Diffie-Hellman key exchange (L7 P5)

(You were given different $g^a \bmod p$ and $b = 3$)
The resulting key should be $\mathbf{(g^a \bmod p)^b}$ since we assume the result > p.

M - (Extra Credits) Q15: SRE to find serial number (L11 P10 - 12 & A3 Q13)

(You were given different .exe) Download the file, disassemble it using a disassembler
Look for the **string near "Please enter serial number" & "Serial number is correct"**.

E - Q16: Linearization attack based on timing work factor (L12 P7 - 9 & A3 Q10)

(You were given different # of characters X with different # of possibilities for each character Y)

Regular exhaustive key search $\mathbf{Y^X / 2}$ on average (need to compute to $\mathbf{2^n}$ **and enter n**);
Using linearization attack: **X * Y / 2** on average.

For each blank, -0.5 if your answer was for the worst case (without / 2) or any other formatting issues.

M - Q17: One-time Pad (L1 P24 & P25 & A1 Q5)

(You were given different encodings for “n” & “o” with different ciphertext)

Encoding for "no": just **concatenate the encoding** for "n" and "o"
Wrong key: **encoding for "no"** ⊕ **ciphertext**

2-pts Short Answers

E - Q18: Feistel cipher (L3 P9 & P11 & P17 & P23 & A1 Q11)

(You were asked 1 out of the following 2)

- DES’s round function does NOT need to be invertible
  Since DES is a Feistel cipher, it’s inherently invertible.
- AES’s round functions need to be invertible
  AES is not a Feistel cipher, therefore, the round functions need to be invertible so that the whole algorithm is invertible.

E - Q19: Defend software attack (L9 P15 - 16 & A3 Q4 Q5 | L11 P13 - 16 & A3 Q9)

(You were asked 1 out of the following 2)

- Defending a stack-based buffer overflow attack
  I. (Mention at least 3) Non-executable stack; canary; ASLR; use safe languages (Java, C#); use safer C functions
  II. (Pick 1 to explain)
    - Use non-executable stack so Trudy can't execute her evil code by overflowing the stack to redirect the return address.
    - Push a canary to the stack. Once canary is overridden, it implies buffer overflowed.
    - Use ASLR to randomize place where code loaded in memory so Trudy can't easily find the address to redirect to.
    - Use safe languages/safer C functions that will throw exception and crash the program if buffer overflowed.
- Mitigating an SRE attack
  I. (Mention at least 3) Anti-disassembly; Anti-debugging; tamper-resistance; code obfuscation
  II. (Pick 1 to explain)
    - Anti-disassembly to confuse static view of code. E.g., encrypt object code or put some junk assembly instructions, etc.
    - Anti-debugging to confuse dynamic view of code. E.g., check IsDebuggerPresent or monitor use of debug registers, etc.
    - Tamper-resistance to make patching harder by detecting if the code is being changed through hash value.
    - Code obfuscation means making code hard to understand so it takes the attacker more time to analyze.

The following questions (Q20 - Q25) may appear in different order on your exam.

H - Q20: Size of the keyspace for a double transposition cipher with m * n matrix (L1 P21)

mPm (permutate all m columns) * nPn (permutate all n rows) = m! * n!

Or more precisely, it's m! * n! - 1 since one of the possibilities is no permutation (original order)

E - Q21: Analyze majority function used in A5/1 (L1 P14 & A1 Q6)

No more than 1 register (zero or 1) step never happen (or 0), since it's impossible to have 0 or 1 bit to be the majority of 3 bits.

E - Q22: Why RC4 is also efficient in software (L2 P17)

Since RC4 operate on byte, not bit.

E - Q23: Decryption in CTR (L4 P11 & A1 Q16)

CTR is similar to stream cipher, that is, E(IV + n, K) can be viewed as the "keystream" to encrypt/decrypt each block

E - Q24: ECC pros & cons (L7 P8)

I. Advantage: smaller key size is needed to achieve the same level of security. Or more secure with the same key size.

II. Since the math is too complex, not everyone can fully understand it to analyze the system (e.g., pick a secure key).

H - Q25: Immune time-based linearization attack on serial number (L12 P7 - 8)

Various ways. For examples:

- Compute and compare the hash values. At least it's more different to time (may depend on the hash function though);
- Set a random timeout to force the computing time to be random (may be too slow);
- Randomize the checking order of each character;
- (Maybe best) Sum the result of bitwise XORing each character. If all characters are the same, the result will remain zero. Code:

```
int main(int argc, const char *argv[]){
        int sum = 0;
        char serial[9]="S123N456\n";
        for(int i = 0; i < 8; i++)
              sum += argv[1][i] ^ serial[i];
        if(sum == 0)
              printf("\nSerial number is correct!\n")
}
```

4-pts Short Answers

E - Q26: RSA (L6 P9 - 10 & A2 Q5)

(You were given different M's and sign or encrypt with the same N = 5 * 7 = 35)

I. e should be relatively prime to $\varphi(N)$ = (5 - 1) * (7 - 1) = 24. So smallest e is 5.

II. $ed \equiv 1 \bmod \varphi(N)$, or $d = e^{-1} \bmod \varphi(N)$, and smallest d is 5 since 5 * 5 mod 24 = 1.

III. To sign (with private key), $[M]_{Bob} = M^d \bmod N = M^5 \bmod 35$
To encrypt (with public key), $\{M\}_{Bob} = M^e \bmod N = M^5 \bmod 35$

The following questions (Q27 - Q28) may appear in different order on your exam.

E - Q27: Kerckhoffs' principle (L1 P11 & A1 Q8)

I. Kerckhoffs' principle: the strength of a cryptosystem depends ONLY on the key (or ONLY key should be hidden).

II. Since secret never remains secret, and we want to be able to find weaknesses before bad guys!

III. For example, A5/1 violates the principle, but then the secret got revealed and now it's broken although the algorithm is complex; Compared to RC4, which follows the principle, is still alive even though the algorithm is simple.

M - Q28: Analyze hash collisions given the length of hash = n (L8 P9 & P11 & A2 Q15)

I. Hashing m messages result in mC2 = m^2 comparisons in total.
And, since length of hash = n, for every 2^n comparisons, we will find a collision.
So we expect to see $m^2 / 2^n$ collisions if we hash m messages.
(No deduction if you answered $m / 2^{n/2}$ WITH explanations)

II. For every 2^n comparisons, we have 1 collision. So, we need $m*2^n$ comparisons to have m collisions.
Suppose we need to compute x hashes, the number of comparisons = ${}_xC_2 = x^2$.
Solving $x^2 = m * 2^n$, we get $x = \sqrt{m * 2^n} = \sqrt{m} * 2^{n/2}$ hashes to find m collisions.
(No deduction if you answered $m * 2^{n/2}$ WITH explanations)