CS 161 Summer 2024 — Introduction to Computer Security — Review: Midterm Lecture

Q1 *Bob's Birthday* **(11 points)**

It's Bob's birthday! Alice wants to send an encrypted birthday message to Bob using ElGamal.

Recall the definition of ElGamal encryption:

- b is the private key, and $B = g^b \bmod p$ is the public key.
- $\mathsf{Enc}(B, M) = (C_1, C_2)$, where $C_1 = g^r \bmod p$ and $C_2 = M \times B^r \bmod p$
- $\mathsf{Dec}(b, C_1, C_2) = C_1^{-b} \times C_2 \bmod p$

Q1.1 (2 points) Mallory wants to tamper with Alice's message to Bob. In response, Alice decides to sign her message with an RSA digital signature. Bob receives the signed message and verifies the signature successfully. Can he be sure the message is from Alice?

- ● Yes, because RSA digital signatures are unforgeable.
- ○ Yes, because RSA encryption is IND-CPA secure.
- ○ No, because Mallory could have blocked Alice's message and replaced it with a different one.
- ○ No, because Mallory could find a different message with the same hash as Alice's original message.

Solution: RSA digital signatures, when paired with a secure hash function, are believed to be unforgeable. See the textbook for a game-based definition of what exactly we mean by unforgeable.

As we discussed in class, ElGamal is malleable, meaning that a man-in-the-middle can change a message in a *predictable* manner, such as producing the ciphertext of the message $2 \times M$ given the ciphertext of M.

Q1.2 (3 points) Consider the following modification to ElGamal: Encrypt as normal, but further encrypt portions of the ciphertext with a block cipher E, which has a block size equal to the number of bits in p. In this scheme, Alice and Bob share a symmetric key K_{sym} known to no one else.

Under this modified scheme, C_1 is computed as $E_{K_{\mathsf{sym}}}(g^r \bmod p)$ and C_2 is computed as $E_{K_{\mathsf{sym}}}(M \times B^r \bmod p)$. Is this scheme still malleable?

- ○ Yes, because block ciphers are not IND-CPA secure encryption schemes
- ○ Yes, because the adversary can still forge $k \times C_2$ to produce $k \times M$
- ● No, because block ciphers are a pseudorandom permutation
- ○ No, because the adversary isn't able to learn anything about the message M

Solution: While block ciphers aren't IND-CPA secure, they are secure when encrypting "random-looking" values because of their properties as pseudorandom permutations. As long as the values you encrypt are unique, the output of the block cipher will always be secure. ElGamal's C_1 and C_2 both appaer random.

Additionally, because block ciphers are a PRP, the scheme is no longer malleable, because modifying the ciphertext in any way causes an unpredictable change to the result of decrypting the block cipher with $D_{K_{\mathsf{sym}}}$.

The remaining parts are independent of the previous part.

For Bob's birthday, Mallory hacks into Bob's computer, which stores Bob's private key b. She isn't able to read b or overwrite b with an arbitrary value, but she can multiply the stored value of b by a random value z known to Mallory.

Mallory wants to send a message to Bob that appears to decrypt as normal, but **using the modified key** $b \cdot z$. Give a new encryption formula for C_1 and C_2 that Mallory should use. Make sure you only use values known to Mallory!

Clarification during exam: For subparts 3 and 4, assume that the value of B is unchanged.

Q1.3 (3 points) Give a formula to produce C_1, encrypting M.

Solution: Mallory should send g^r with some randomly chosen r, as usual.

Q1.4 (3 points) Give a formula to produce C_2, encrypting M.

Solution: Mallory should send $C_2 = m \times B^{rz} \mod p$.

Q2 *Cryptography: EvanBot Signature Scheme* **(12 points)**

EvanBot decides to make a signature scheme!

To initialize the system, a Diffie-Hellman generator g and prime p are generated and shared to all parties. The private key is some $x \bmod p$ chosen randomly, and the public key is $y = g^x \bmod p$.

To sign a message m such that $2 \leq m \leq p-2$:

1. Choose a random integer k between 2 and $p-2$.
2. Set $r = g^k \bmod p$.
3. Set $s = (\mathsf{H}(m) - xr)k^{-1} \bmod (p-1)$. If $s = 0$, restart from Step 1.
4. Output (r, s) as the signature.

Clarification after exam: k is chosen to be coprime to $p-1$.

To verify, check that $g^{H(m)} \equiv$ _____ $\bmod p$. We will fill in this blank in the next few subparts.

Q2.1 (3 points) Select the correct expression for $H(m)$ in terms of x, r, k, s and $p-1$.
HINT: Use Step 3 of the signature algorithm.

- ○ $k(xr)^{-1} + s \bmod (p-1)$
- ○ $k^{-1} + xr \bmod (p-1)$
- ○ $ks - xr \bmod (p-1)$
- ● $ks + xr \bmod (p-1)$

Solution: From step 3:

$$\begin{aligned} s &\equiv (\mathsf{H}(m) - xr)k^{-1} \bmod (p-1) \\ sk &\equiv \mathsf{H}(m) - xr \bmod (p-1) \\ sk + xr &\equiv H(m) \bmod (p-1) \\ H(m) &\equiv ks + xr \bmod (p-1) \end{aligned}$$

Q2.2 (4 points) Using the previous result, select the correct value for the blank in the verification step. *HINT: Replace the $H(m)$ in $g^{H(m)}$ with your results from the previous subpart.*

- ○ $y^s r^2 \bmod p$
- ○ $r^y r^s \bmod p$
- ● $y^r r^s \bmod p$
- ○ $r g^{yr} \bmod p$

Solution:

$$\begin{aligned} & g^{ks+xr} \bmod p \\ & \equiv g^{ks} \cdot g^{xr} \bmod p \\ & \equiv (g^k)^s \cdot (g^x)^r \bmod p \\ & \equiv r^s y^r \equiv y^r r^s \bmod p \end{aligned}$$

Q2.3 (5 points) Show how to recover the private key x if a signature is generated such that $s = 0$ (i.e. the check on Step 3 is ignored).

Solution: If $s = 0$, then $0 \equiv (H(m) - xr)k^{-1} \bmod (p-1)$ per Step 3, which means $xr = H(m) \bmod (p-1)$ and we can solve for x. Note that k is implicitly coprime to $p-1$ by construction in the protocol.

Q3 *The Lorenzo Von Matterhorn* **(0 points)**

Barney needs to make sure that no attackers can access his highly sensitive, top secret playbook tricks!

For each password scheme, select all true statements. Assume that:

- Each user has a unique username, but not necessarily a unique password.
- All information is stored in a read-only database that both the server and the attacker can access.
- The server has a symmetric key K not known to anyone else. The server also has a secret key SK not known to anyone else, and a corresponding public key PK that everyone knows.
- An *operation* is defined as one of the following actions: hash, encryption, decryption, and HMAC.
- The attacker does not have access to a client UI; therefore, online attacks are not possible.

Q3.1 For each user, the database contains $\mathsf{username}$ and $\mathsf{H}(\mathsf{password})$, where H is a cryptographic hash function.

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [x] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [ ] The server can list all plaintext passwords by computing at most one operation per user
- [x] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can hash the password to check that it matches the hash in the database.

B: The attacker can hash the password (hashes aren't keyed) and check that it matches the hash in the database.

C: The server cannot compute one operation to reverse a hash.

D: The attacker can conduct an offline brute-force attack, hashing every possible password and comparing to the hashes in the database.

Q3.2 For each user, the database contains username and $\mathsf{HMAC}(K, \mathsf{password})$.

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [ ] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [ ] The server can list all plaintext passwords by computing at most one operation per user
- [ ] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can HMAC the password to check that it matches the HMAC in the database.

B: An attacker cannot compute the output of HMAC without knowing the key input K.

C: The server cannot compute one operation to reverse HMAC.

D: The attacker cannot compute HMACs without knowing the key input K.

Q3.3 For this subpart, Enc denotes an IND-CPA secure symmetric encryption function.

For each user, the database contains username and $\mathsf{Enc}(K, \mathsf{password})$.

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [ ] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [x] The server can list all plaintext passwords by computing at most one operation per user
- [ ] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can decrypt the password in the database to check that it matches the password given by the user.

B: An attacker cannot decrypt passwords without knowing the key input K.

C: The server can decrypt the password in the database.

D: An attacker cannot decrypt passwords without knowing the key input K.

Q3.4 For this subpart, RSA denotes RSA encryption without OAEP padding.

For each user, the database contains username and RSA(PK, password).

- [x] If a user inputs a username and password, the server can verify whether the password is correct
- [x] Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- [x] The server can list all plaintext passwords by computing at most one operation per user
- [x] An attacker can list all passwords by computing at most one operation per possible password
- [ ] None of the above

Solution:

A: The server can encrypt the password to check that it matches the password in the database.

B: The attacker can also encrypt passwords, because everybody knows the public key.

C: The server can decrypt the password in the database.

D: An attacker can encrypt every possible password and compare the encryptions to the encryptions in the database.

Q4 *Ra's Al Gamal* **(0 points)**

Recall the ElGamal scheme from lecture:

- KeyGen() $= (b, B = g^b \bmod p)$
- Enc$(B, M) = (C_1 = g^r \bmod p, C_2 = B^r \times M \bmod p)$

Q4.1 Is the ciphertext (C_1, C_2) decryptable by someone who knows the private key b? If you answer yes, provide a decryption formula. You may use C_1, C_2, b, and any public values.

● Yes ○ No

Solution: The decryption formula is $M = C_1^{-b} \times C_2$.

Q4.2 Consider an adversary that can efficiently break the discrete log problem. Can the adversary decrypt the ciphertext (C_1, C_2) without knowledge of the private key? If you answer yes, briefly state how the adversary can decrypt the ciphertext.

● Yes ○ No

Solution: An adversary that can break the discrete log problem can recover r from $C_1 = g^r$ or b from $B = g^b$, so they can compute g^{br} and recover the original message.

Q4.3 Consider an adversary that can efficiently break the Diffie-Hellman problem. Can the adversary decrypt the ciphertext (C_1, C_2) without knowledge of the private key? If you answer yes, briefly state how the adversary can decrypt the ciphertext.

● Yes ○ No

Solution: An adversary that can break the Diffie-Hellman problem can recover g^{br} from $C_1 = g^r$ and $B = g^b$, so they can recover the original message.

Q5 *PRNGs and Diffie-Hellman Key Exchange* **(15 points)**

Eve is an eavesdropper listening to an insecure channel between Alice and Bob.

1. Alice and Bob each seed a PRNG with different random inputs.
2. Alice and Bob each use their PRNG to generate some pseudorandom output.
3. Eve learns both Alice's and Bob's pseudorandom outputs from step 2.
4. Alice, without reseeding, uses her PRNG from the previous steps to generate a, and Bob, without reseeding, uses his PRNG from the previous steps to generate b.
5. Alice and Bob perform a Diffie-Hellman key exchange using their generated secrets (a and b). Recall that, in Diffie-Hellman, neither a nor b are directly sent over the channel.

For each choice of PRNG constructions, select the minimum number of PRNGs Eve needs to compromise (learn the internal state of) in order to learn the Diffie-Hellman shared secret $g^{ab} \bmod p$. Assume that Eve always learns the internal state of a PRNG between steps 3 and 4.

Q5.1 (3 points) Alice and Bob both use a PRNG that outputs the same number each time.

● (A) Neither PRNG ○ (C) Both PRNGs ○ (E) ——

○ (B) One PRNG ○ (D) Eve can't learn the secret ○ (F) ——

Solution: Eve observes the PRNG outputs. Since both PRNGs output the same number each time, Eve also learns the values of a and b. She can use this to compute the shared secret $g^{ab} \bmod p$ without compromising any PRNGs.

Q5.2 (3 points) Alice uses a secure, rollback-resistant PRNG. Bob uses a PRNG that outputs the same number each time.

● (G) Neither PRNG ○ (I) Both PRNGs ○ (K) ——

○ (H) One PRNG ○ (J) Eve can't learn the secret ○ (L) ——

Solution: Eve observes Bob's PRNG output and learns the value of b. Alice will send $g^a \bmod p$ in his half of the exchange. Eve can compute $(g^a)^b \bmod p$ to learn the shared secret without compromising any PRNGs.

Q5.3 (3 points) Alice and Bob both use a secure, rollback-resistant PRNG.

◯ (A) Neither PRNG ◯ (C) Both PRNGs ◯ (E) ——

● (B) One PRNG ◯ (D) Eve can't learn the secret ◯ (F) ——

Solution: Eve only needs to compromise one PRNG to learn one of the secrets. For example, if Eve compromises Alice's PRNG, she learns a and can compute $(g^b)^a \bmod p$ to learn the shared secret (because Bob sends $g^b \bmod p$ in his half of the exchange). Alternatively, if Eve compromises Bob's PRNG, she learns b and can compute $(g^a)^b \bmod p$ to learn the shared secret (because Alice sends $g^a \bmod p$ in her half of the exchange).

For the rest of the question, consider a different sequence of steps:

1. Alice and Bob each seed a PRNG with different random inputs.
2. Alice uses her PRNG from the previous step to generate a, and Bob uses his PRNG from the previous step to generate b.
3. Alice and Bob perform a Diffie-Hellman key exchange using their generated secrets (a and b).
4. Alice and Bob, without reseeding, each use their PRNG to generate some pseudorandom output.
5. Eve learns both Alice's and Bob's pseudorandom outputs from step 4.

As before, assume that Eve always learns the internal state of a PRNG between steps 3 and 4.

Q5.4 (3 points) Alice and Bob both use a secure, but not rollback-resistant PRNG.

◯ (G) Neither PRNG ◯ (I) Both PRNGs ◯ (K) ——

● (H) One PRNG ◯ (J) Eve can't learn the secret ◯ (L) ——

Solution: Because there is no rollback resistance, if Eve compromises one PRNG, Eve can deduce previous PRNG output and learn a secret (either a or b), which is enough to compute the shared secret (as in the previous part).

Q5.5 (3 points) Alice and Bob both use a secure, rollback-resistant PRNG.

◯ (A) Neither PRNG ◯ (C) Both PRNGs ◯ (E) ——

◯ (B) One PRNG ● (D) Eve can't learn the secret ◯ (F) ——

Solution: Even if Eve compromises both PRNGs, because they are rollback-resistant, Eve cannot deduce the secrets a and b (i.e. previous PRNG output).