

Homework 3

Encryption

Botond Lajos Perényi

Task 1: RSA cryptosystem

The paper claims that with having $\phi(n)$ it is trivial to break RSA. But it does not tell anything about how one might gain $\phi(n)$, which is not easy for large numbers, that are generally used with RSA. Finding the Totient function ($\phi(n)$) generally requires to calculate all the primes that are smaller than n , which is a time consuming task.

My advice based on this paper alone would be to keep using the already existing encryption system.

Task 2: Security definitions

Adversary can send n as one message (m_1), whereas the other message (m_2) should not be divisible by n . This way, if the ciphertext is 0, they can know, that m_1 was encrypted. This is due to the fact that remainders do not change with exponentiation.

Task 3: ElGamal encryption

$Enc(pk, m) :$

$$y \leftarrow 1, \dots, q-1$$

$$s = h^y \in \mathbb{G}$$

$$E(pk, m) = (c_1, c_2) = (g^y, m \cdot s) \in \mathbb{G} \times \mathbb{G}$$

Part i Create a re-randomized version of ElGamal:

$$y' \leftarrow 1, \dots, q-1 \quad \text{We generate a new randomness}$$

$$s' = h^{y'} \in \mathbb{G}$$

Then using the new randomness, we re-randomize the original message:

$$(c_1', c_2') = (g^y \cdot g^{y'}, m \cdot h^y \cdot h^{y'}) = (g^{y+y'}, m \cdot h^{y+y'})$$

Note that the new randomness is added on to the original one.

Part ii We can use the method previously described to create a new c' with different randomness, that is valid under the same ElGamal scheme. Then we can give this new c' to decrypt, and we will get back the original message, as it was not altered (only the randomness was changed).

Part iii With $c_{11} = c_{12}$ we can deduce that the same randomness (y) was chosen, because $c_1 1 = g^y$, where g is fixed for the scheme. Thus we can write the ciphertexts quotient as:

$$\frac{c_{21}}{c_{22}} = \frac{m_1 \cdot h^y}{m_2 \cdot h^y} = \frac{m_1}{m_2}$$

This way we know the ratio of the two messages, but we can't know their exact value.

Part iv ElGamal's homomorphic property:

Useful: In electronic voting schemes. To tally the votes one can multiply the messages together to get a count of the votes. (For this application one needs to choose primes as message values.)

Violates the security: Messaging over insecure channel. An outsider can capture and forge new (potentially meaningful) messages without decryption.

Task 4: Attack on RSA encryption

We have two captured ciphertexts of the same message:

$$\begin{array}{ll} m & m \\ c_1 = 7 & c_2 = 16 \\ pk_1 = (3, 57) & pk_2 = (5, 57) \end{array}$$

Writing up the messages:

$$7 = m^3 \pmod{57} \qquad 16 = m^5 \pmod{57}$$

We can notice that the base for both systems are the same. Thus we can use the Chinese Remainder Theorem to make breaking the message, using the fact that $57 = 3 \cdot 19$, where both factors are prime. First find m in both remainders:

Mod 3

$$\begin{array}{ll} 7 = m^3 \pmod{3} & \rightarrow 1 = m^3 \pmod{3} \\ 16 = m^5 \pmod{3} & \rightarrow 1 = m^3 \pmod{3} \end{array}$$

Then check the potential values: Thus we get $m = 1 \pmod{3}$.

m	m^3	m^5
0	0	0
1	1	1
2	2	2

Mod 19

$$\begin{array}{ll} 7 = m^3 \pmod{19} & \rightarrow 7 = m^3 \pmod{19} \\ 16 = m^5 \pmod{19} & \rightarrow 16 = m^3 \pmod{19} \end{array}$$

Then check the potential values:

m	m^3	m^5
0	0	0
1	1	
2	8	
3	8	
4	7	17
5	11	
6	7	5
7	1	
8	18	
9	7	16
10	12	
11	1	
12	18	
12	12	
14	8	
15	12	
16	11	
17	11	
18	18	

I only calculated those that are of interest for the m^5 column. We can see, that the result is $m = 9 \pmod{19}$

Combining the results we get:

$$m = 1 \pmod{3}$$

$$m = 9 \pmod{19} \rightarrow m = 19k + 9$$

Rewriting it to $\pmod{3}$:

$$19k + 9 = 1 \pmod{3} \rightarrow 1k + 0 = 1 \pmod{3}$$

$$k = 1 \pmod{3}$$

$$k = 3t + 1$$

Writing the result back to m :

$$m = 19k + 9 = 19(3t + 1) + 9 = 57t + 28$$

Thus we arrive to $m = 28$. (We have $\pmod{57}$, so the t part can be ignored.)

Task 5: Choosing cryptographic primitives

For this task, I used Claude's Sonnet model.**claude3** This is the result:

Cryptographic Solutions for Security Requirements

Let me provide cryptographic schemes for each of your security requirements while ensuring we use strong primitives resistant to common attacks.

1. Data Confidentiality and Integrity in Transmission

Recommended Schemes:

- **Confidentiality:** AES-GCM (Galois/Counter Mode)
- **Key Exchange:** Diffie-Hellman for session key establishment

Key Length Recommendations:

- AES: 256-bit keys
- Diffie-Hellman: 2048-bit or 3072-bit parameters

Justification: AES-GCM provides both encryption (confidentiality) and authentication (integrity) in a single efficient operation. The 256-bit key length for AES offers a strong security margin against brute force attacks. For key exchange, Diffie-Hellman with at least 2048-bit parameters provides sufficient security against computational attacks.

2. Data Confidentiality and Integrity at Rest

Recommended Schemes:

- **Encryption:** AES in GCM mode
- **Key Management:** RSA for key wrapping/storage

Key Length Recommendations:

- AES: 256-bit keys
- RSA: 3072-bit keys

Justification: For data at rest, AES-GCM continues to be an excellent choice as it provides both confidentiality and integrity verification. RSA with 3072-bit keys offers comparable security to AES-256 for key management operations. This combination ensures data remains protected even when stored for long periods.

3. Strong Authentication Mechanisms

Recommended Schemes:

- **Digital Signatures:** RSA or ElGamal for entity authentication
- **Hash Functions:** SHA-256 for password hashing (with appropriate salting)

Key Length Recommendations:

- RSA: 3072-bit keys
- ElGamal: 3072-bit keys
- SHA-256: 256-bit output

Justification: Digital signatures using RSA or ElGamal with 3072-bit keys provide strong authentication of users and services. For password-based authentication, SHA-256 combined with proper salting techniques provides resistance against dictionary attacks and rainbow tables.

4. Resistance to Common Attacks

For Side-Channel Attack Resistance:

- Implement constant-time operations for cryptographic implementations
- Use ChaCha20-Poly1305 as an alternative to AES in environments where side-channel resistance is critical
- Apply blinding techniques for RSA operations

For Brute Force Attack Resistance:

- The recommended key lengths (AES-256, RSA-3072, DH-3072) provide sufficient security margins
- Implement rate limiting and account lockout mechanisms

General Recommendations:

- Avoid deprecated algorithms like MD4, MD5, and SHA-1
- Use BLAKE2 instead of older hash functions when possible
- Implement proper key rotation policies
- Use hardware security modules (HSMs) for critical key operations

The recommended schemes and key lengths provide a security level equivalent to approximately 128-bit symmetric security, which is considered secure against attacks by classical computers for the foreseeable future. For protection against quantum computing threats, consider implementing quantum-resistant algorithms alongside these classical approaches.

My evaluation:

Using AES-GCM for encryption and authentication is correct, and the recommended 256-bit keylength is the highest in the standard, widely regarded as the most secure form of AES. Diffie-Hellman for key exchange is also a common standard, and the recommended keylength is also sufficient. (For this I used the Federal Office for Information Security's report **bsi**) For *Data Confidentiality and Integrity at Rest* it makes a bit higher recommendation, but overall remains the same. As far as I can tell, it is common practice to encrypt data at rest using AES. It is also sensible to use the same method for both sending and storing the data, as it makes more efficient use of the algorithms.

The authentication isn't exactly clear, it recommends two options. Both are accepted as good solutions, and are used for this purpose. For the keylength it recommends the maximums from the standards. This is good for future proofing, but might not be necessary at the moment. (Shorter keys can provide similar level of protection.)

Recommendations for *Resistance to Common Attacks* are the usual recommendations that can be found on most places. I would note that it is interesting that it both states that *AES-256, RSA-3072, DH-3072 provide sufficient security margins* and then recommends to still implement more defensive mechanisms. (I would implement those mechanisms however.) It brings up MD4, MD5 and SHA-1 most likely because I prepared the model with the list of encryption standards we learned at class.