CISC 468: CRYPTOGRAPHY

LESSON 12: THE ELGAMAL ENCRYPTION SCHEME

Furkan Alaca

READINGS

- Section 8.4: Security of the Diffie-Hellman Key Exchange, Paar & Pelzl
- Section 8.5: The Elgamal Encryption Scheme, Paar & Pelzl

INTRODUCTION

- Elgamal encryption can be viewed as an extension of the DHKE protocol
- It is also based on the intractability of the discrete logarithm problem (which we saw last week) and the Diffie-Hellman problem (which we will define today)

DHKE REVIEW

0. Setup: Choose a large prime p and an integer $\alpha \in \{2, 3, \ldots, p-2\}$, and publish (p, α).
1. Alice selects a private key $a \in \{2, 3, \ldots, p-2\}$, computes $A = \alpha^a \mod p$, and sends A to Bob.
2. Bob selects a private key $b \in \{2, 3, \ldots, p-2\}$, computes $B = \alpha^b \mod p$, and sends B to Alice.
3. Alice computes $k_{AB} = B^a = (\alpha^b)^a \mod p$.
4. Bob computes $k_{AB} = A^b = (\alpha^a)^b \mod p$.
5. Alice and Bob initiate secure communication using k_{AB}, e.g., as a symmetric key for encryption.

DHKE REVIEW: EXAMPLE

Using domain parameters $p = 29, \alpha = 2$:

$$
\begin{array}{ccc}
\textbf{Alice} & & \textbf{Bob} \\
\text{choose } a = k_{pr,A} = 5 & & \text{choose } b = k_{pr,B} = 12 \\
A = k_{pub,A} = 2^5 \equiv 3 \bmod 29 & & B = k_{pub,B} = 2^{12} \equiv 7 \bmod 29 \\
 & \xrightarrow{A=3} & \\
 & \xleftarrow{B=7} & \\
k_{AB} = B^a \equiv 7^5 = 16 \bmod 29 & & k_{AB} = A^b = 3^{12} \equiv 16 \bmod 29
\end{array}
$$

SECURITY OF DHKE: PASSIVE ATTACKS

- Passive attacker knows public parameters (p, α) and can eavesdrop A and B
- Given two elements $A = \alpha^a$ and $B = \alpha^b$ in a finite cyclic group G of order n, and a primitive element $\alpha \in G$, the *Diffie-Hellman problem* (DHP) is to find the group element α^{ab}

DHP VS DLP

If an efficient solution to the discrete logarithm problem (DLP) was known, an attacker could:

1. Compute Alice's private exponent $a \equiv \log_{\alpha} A \mod p$.
2. Compute the key $k_{AB} \equiv B^{a} \mod p$.

The DLP is infeasible if p is sufficiently large.

SECURITY OF DHKE: PARAMETER REQUIREMENTS

- For 112-bit security, the prime p must be 2048 bits
  - For 128-bit security, p must be 3072 bits (but 4096-bit may have better compatibility)
- The largest prime factor of $p - 1$ must be at least 256 bits for 128-bit security
  - See Pohlig-Hellman attack (Section 8.3.3) if interested

SECURITY OF DHKE: SUBGROUP CONFINEMENT ATTACK

- Assume a prime p and primitive element α are chosen
- Alice computes $A = \alpha^a \mod p$ and sends it to Bob
  - But Eve intercepts A, computes A^k and sends that to Bob instead
- Bob computes $B = \alpha^b \mod p$ and sends it to Alice
  - But Eve intercepts B, computes B^k and sends that to Alice instead
- If k is carefully chosen by the attacker, Alice and Bob will compute a k_{AB} that is an element of a small subgroup of $\mathbb{Z}_P^*$ that can be exhaustively searched

SECURITY OF DHKE: ACTIVE ATTACKS

- The subgroup confinement attack and the person-in-the-middle attacks are *active attacks*
- "Plain" DHKE is not secure against active attacks
- Defenses against active attacks:
  - Use known safe parameters, e.g., from RFC 7919, to avoid the chance of using weak groups, and perform the recommended checks to ensure the other party is not confining your client to a small subgroup
  - Perform integrity checks to ensure that an attacker has not modified any messages in transit (we will learn how, with digital signatures)

ELGAMAL: BASIC MECHANISM

Basic mechanism by which Alice sends a message x to Bob:

1. Bob executes the DHKE set-up protocol to select a large prime p and primitive element α.
2. Alice and Bob perform a DHKE to derive a shared key k_M.
3. Alice uses k_M as a multiplicative mask to encrypt x by computing $y \equiv x \cdot k_M \mod p$.
4. Bob decrypts the message by computing $x \equiv y \cdot k_M^{-1} \mod p$.

ELGAMAL SET-UP PHASE

- The set-up phase is executed by the party who will receive the message
- The receiver chooses a large prime p and a primitive element α, and publishes them (e.g., on their website)
  - As with RSA, p should be at least 2048 bits and can be generated using an appropriate prime-finding algorithm
- The receiver selects a random private key d and public key $\beta = \alpha^d \mod p$
  - Same process as DHKE
  - This key pair does not change (i.e., is used repeatedly)

ELGAMAL ENCRYPTION PHASE

- The sender must generate a new public-private key pair i and $k_E \equiv \alpha^i \mod p$ for the encryption of every message
  - E denotes "ephemeral" (existing only temporarily)
  - Ensures that Elgamal is a probabilistic encryption scheme
- The sender computes $k_M \equiv \beta^i \mod p$ and the ciphertext $y \equiv x \cdot k_M \mod p$
  - Property of cyclic groups: each x maps to unique ciphertext
  - If k_M is randomly drawn from $\mathbb{Z}_p^*$, every y is equally likely
- The sender sends k_E along with y to the receiver
  - Thus, the ciphertext (k_E, y) is twice as long as the message x

ELGAMAL DECRYPTION PHASE

- Receiver computes masking key $k_M = (k_E)^d$
- Receiver recovers original plaintext $x = y \cdot k_M^{-1}$

ELGAMAL ENCRYPTION: EXAMPLE

Alice		**Bob**
message $x = 26$		generate $p = 29$ and $\alpha = 2$ choose $k_{pr,B} = d = 12$ compute $\beta = \alpha^d \equiv 7 \bmod 29$
	$\xleftarrow{k_{pub,B}=(p,\alpha,\beta)}$	
choose $i = 5$ compute $k_E = \alpha^i \equiv 3 \bmod 29$ compute $k_M = \beta^i \equiv 16 \bmod 29$ encrypt $y = x \cdot k_M \equiv 10 \bmod 29$		
	$\xrightarrow{y,k_E}$	
		compute $k_M = k_E^d \equiv 16 \bmod 29$ decrypt $x = y \cdot k_M^{-1} \equiv 10 \cdot 20 \equiv 26 \bmod 29$

ELGAMAL PROOF OF CORRECTNESS

Proof that decrypting the ciphertext yields the original plaintext:

$$\begin{aligned} d(k_E, y) &\equiv y \cdot (k_M)^{-1} \mod p \\ &\equiv [x \cdot k_M] \cdot (k_E^d)^{-1} \mod p \\ &\equiv [x \cdot (\alpha^d)^i][(\alpha^i)^d]^{-1} \mod p \\ &\equiv x \cdot \alpha^{d \cdot i - d \cdot i} \mod p \\ &\equiv x \mod p \end{aligned}$$

COMPUTATIONAL ASPECTS

- Modular exponentiation is used in key generation, encryption, and decryption, so square-and-multiply algorithm is used (as we covered for RSA)
- The exponentiations required for encryption are independent of the plaintext—they can be precomputed when CPU load is low, and stored for when encryption is needed
- For decryption, computing $k_M = k^d \mod p$ followed by the inverse k_M^{-1} can be combined into one step using Fermat's Little Theorem (see Section 8.5.3)

SECURITY OF ELGAMAL: KEY SIZE REQUIREMENTS

- Passive attacker can learn $p, \alpha, \beta = \alpha^d, k_E = \alpha^i, y = x \cdot \beta^i$
- If the attacker can compute DLPs, they may compute either:
  - $d = \log_\alpha \beta \bmod p$ followed by $x \equiv y \cdot (k_E^d)^{-1} \bmod p$

    or
  - $i = \log_\alpha k \bmod p$ followed by $x \equiv y \cdot (\beta^i)^{-1} \bmod p$
- To ensure that this is infeasible, p should be at least 2048 bits

SECURITY OF ELGAMAL: ACTIVE ATTACKS

- Just as with DHKE, take necessary precautions against active attacks, e.g., small subgroup confinement or person-in-the-middle attacks
- Active attacks against Elgamal can also exploit:
  - Ephemeral key reuse
  - Malleability of Elgamal

SECURITY OF ELGAMAL: EXPLOITING EPHEMERAL KEY REUSE

- Assume Alice uses the same secret exponent i to encrypt two messages x_1 and x_2
  - In this case, both masking keys would be $k_M = \beta^i$
  - Alice would send (y_1, k_E) and (y_2, k_E) over the channel
- If Oscar figures out (e.g., guesses) the first message x_1, he can compute the masking key as $k_M \equiv y_1 x_1^{-1} \mod p$ and then decrypt the second message by computing $x_2 \equiv y_2 k_M^{-1} \mod p$
- Appropriate defense against this attack is to randomly select i and to ensure that the same value is not reused

SECURITY OF ELGAMAL: EXPLOITING MALLEABILITY

- Similarly to "Schoolbook RSA", ciphertext generated by "Schoolbook Elgamal" is susceptible to manipulation
- If Oscar intercepts the ciphertext (k_E, y), he can replace it with (k_E, sy) where s is some integer
- The modified ciphertext would then be decrypted by the receiver to sx (as an exercise, you may verify this)
  - For instance, the attacker can choose $s = 2$ to double the value of a money transfer
- Appropriate defense against this attack is to use padding, similar to what is done with RSA

RECAP

- Elgamal implementations are widely available, including in free software such as GnuPG and OpenSSL
- We consider the Elgamal encryption scheme over the group $\mathbb{Z}_p^*$ where p is prime
  - But it can be applied to other cyclic groups too where the discrete logarithm problem and Diffie-Hellman problem are intractable