Justin Ciocoi Nov. 16, 2023

CSCI 360 Textbook Notes

Chapter 8: Public-Key Cryptosystems Based on the Discrete Logarithm Problem

8.1: Diffie-Hellman Key Exchange

- The *Diffie-Hellman Key Exchange* was the first asymmetric cryptographic scheme that was published in open literature
- It provides a solution to the key distribution problem by allowing two parties to derive a common secret key over an insecure channel
- The basic idea behind DHKE is that exponentiation in $\mathbb{Z}_p^*$, where p is prime is a one way function and that exponentiation is commutative:

$$k = (\alpha^x)^y \equiv (\alpha^y)^x \ mod \ p$$

- The value k is the joint secret which can be used as the session key between the two parties
- We will now consider how the DHKE protocol works over $\mathbb{Z}_p^*$ where two people, *Person A*, and *Person B* would like to establish a shared secret key
- There are two steps to the DHKE protocol
    - **Diffie-Hellman Set-Up**
        1. Choose a large prime, p
        2. Choose an integer, $\alpha \in \{2, 3, 4, ..., p - 2\}$
        3. Publish p and α
    - **Diffie-Hellman Key Exchange**

Diffie–Hellman Key Exchange

Alice		**Bob**
choose $a = k_{pr,A} \in \{2,\dots,p-2\}$		choose $b = k_{pr,B} \in \{2,\dots,p-2\}$
compute $A = k_{pub,A} \equiv \alpha^a \bmod p$		compute $B = k_{pub,B} \equiv \alpha^b \bmod p$
	$\xrightarrow{k_{pub,A}=A}$	
	$\xleftarrow{k_{pub,B}=B}$	
$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$		$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$

- Here, *Alice* computes $B^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \; mod \; p$
- *Bob* computes $A^b \equiv (\alpha^a)^b \equiv \alpha^{ab} \; mod \; p$
- And thus they share the key $k_{AB} \equiv \alpha^{ab} \; mod \; p$

8.2: Some Algebra

- **8.2.2: Cyclic Groups**
  - In cryptography, we are almost always concerned with finite structures
  - **Definition 8.2.2**: *Finite Group*

    A group, $(G, \circ)$ is finite if it has a finite number of elements. we denote the *cardinality* or *order* of the group G using $|G|$
  - **Definition 8.2.3**: *Order of an element*

    The *order*, $ord(a)$ of an element, a, in group $(G, \circ)$ is the smallest positive integer, k such that:

    $$a^k = a \circ a \circ a \circ ... \circ a \; (k \; times) = 1$$

    where 1 is the identity element of G
  - **Definition 8.2.4**: *Cyclic Group*

    A group, G, which contains an element α, with a maximum order, $ord(a) = |G|$, is said to be cyclic

    Elements with maximum order are called *primitive elements* or *generators*
  - An element α of a group G is called a generator when every element, a, in the set can be written as a power of $\alpha^i = a$ where i is also in the group
  - There are several interesting properties of cyclic groups, and the most important ones for cryptographic applications are as follow:

- **Theorem 8.2.2:** For every prime p, the group ($\mathbb{Z}_p^*, \cdot$) is an abelian finite cyclic group

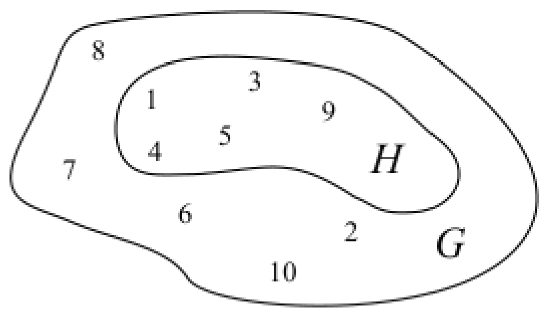
  This essentially means that every prime field is cyclic, which is a fact with far reaching consequences in cryptography

- **Theorem 8.2.3:** Let G be a finite group. Then for every $a \in G$, it holds that:
  - $a^{|G|} = 1$
  - $|G| \; mod \; order(a) = 0$
- Theorem **8.2.4:** Let G be a finite cyclic group; then it holds that:
  - The number of primitive elements of G is $\Phi(|G|)$
  - If $|G|$ is prime, then all elements $a \neq 1 \in G$ are primitive

- **8.2.3: Subgroups**
  - Now we discuss subsets of groups which are groups themselves, also referred to as *subgroups*
  - In the case of cyclic groups, there is an easy way to generate subgroups, which comes from the following theorem
  - **Theorem 8.2.5:** *Cyclic Subgroup Theorem*

    Let $(G, \circ)$ be a cyclic group

    Then, every element with $a \in G$ with $order(a) = s$ is the primitive element of a cyclic subgroup with s elements
  - This essentially tells us that any element of a cyclic group is the generator of a subgroup which is, in turn, also cyclic
  - Below we can see an illustration of group G, enclosing its subgroup, H



- **Theorem 8.2.6:** *Lagrange's Theorem*

  Let H be a subgroup of G. Then, $|H|$ evenly divides $|G|$

- **Theorem 8.2.7:**

  Let G be a finite cyclic group of order n and let α be a generator of G

  Then, for every integer k that divides n evenly, there exists exactly one cyclic subgroup H of G of order k

  This subgroup is generates by $\alpha^{n/k}$

  H consists exactly of the elements $a \in G$ which satisfy the condition $a^k = 1$

  There are no other subgroups

  8.3: The Discrete Logarithm Problem

- **8.3.1: The Discrete Logarithm Problem in Prime Fields**
  - **Definition 8.3.1:** *Discrete Logarithm Problem (DLP) in* $\mathbb{Z}_p^\star$

    Given the finite cyclic group $\mathbb{Z}_p^*$ of order $p-1$, a primitive element $\alpha \in \mathbb{Z}_p^*$, and another element $\beta \in \mathbb{Z}_p^*$

    The DLP is the problem of determining the integer $1 \leq x \leq p-1$ such that

    $$\alpha^x \equiv \beta \; mod \; p$$

  - We can also formally write that

    $$x = log_\alpha \beta \; mod \; p$$

- **8.3.2: The Generalized Discrete Logarithm Problem**
  - The feature which makes the DLP particularly useful in cryptography is that it is not restricted to the multiplicative group $\mathbb{Z}_p^*$, where p is prime, but can instead be defined over any cyclic groups
  - This is called the *generalized discrete logarithm problem* and can be stated as follows
  - **Definition 8.3.2:** *Generalized Discrete Logarithm Problem*

    Given a finite cyclic group G with the group operation $\circ$ and cardinality n

    We consider a primitive element $\alpha \in G$ and another element $\beta \in G$

The discrete logarithm problem is finding the integer x, where $1 \leq x \leq n$ such that:

$$\beta = \alpha \circ \alpha \circ \alpha \circ ... \circ \alpha(x \; times) = \alpha^x$$

- **8.3.3: Attacks Against the Discrete Logarithm Problem**
  - *Brute-Force Search*
    - In a brute-force attack, we can simply compute powers of the generator α successively until the result equals β i.e.

$$\begin{aligned} \alpha^1 &= \beta \\ \alpha^2 &= \beta \\ \alpha^3 &= \beta \\ &\vdots \\ \alpha^x &= \beta \end{aligned}$$

  - *Shanks' Baby-Step Giant-Step Method*
    - Shanks' algorithm is a time-memory tradeoff method, which reduces the overall time it will take to run a brute-force algorithm at the cost of extra storage
    - The idea is based on rewriting the discrete logarithm $x = log_\alpha \beta$ in a two-digit representation as follows:

$$x = x_g m + x_b \text{ for } 0 \leq x_g, \; x_b < m$$

    - The value m, here is chosen to be the square root of the group order, i.e. $m = \sqrt{|G|}$
    - Now, the discrete logarithm can be written as $\beta = \alpha^x = \alpha^{x_g m + x_b}$ which leads to:

$$\beta \cdot (\alpha^{-m})^{x_g} = \alpha^{x_b}$$

    - The core idea here is that this equation can be solved by searching for x_g and x_b separately, i.e. using a divide-and-conquer approach
    - In the first phase, or *baby-step phase* we compute and store all values α^{x_b} where $0 \leq c_b < m$
      - This phase requires $m \approx \sqrt{|G|}$ steps, or group operations, and needs to store $m \approx \sqrt{|G|}$ group elements
    - In the *giant-step phase*, the algorithm checks for all x_g in the range $0 \leq x_g < m$ whether the following condition is fulfilled

$$\beta \cdot (\alpha^{-m})^{x_g} = \alpha^{x_b}$$

for some stored entry α^{x_b} which was computed in the prior phase

- If a match for some pair $(x_{g,0}, x_{b,0})$ exists, then the discrete logarithm is given by

$$x = x_{g,0}m + x_{b,0}$$

- This method requires $O(\sqrt{|G|})$ steps and an equal amount of memory
- In a group of order 2^{80}, an attacker would only need approximately $2^{40} = \sqrt{2^{80}}$ computations and memory locations, which is easily achievable with todays computer systems and storage mediums
- Thus, in order to obtain an attack complexity of 2^{80}, a group must have a cardinality of at least 2^{160}
    - In the case of groups $G = \mathbb{Z}_p^*$ the prime p should have a length of at least 160 bit

- *Pollard's Rho Method*
    - In Pollard's rho method, the runtime is approximately the same as Shanks' algorithm, but it has only negligible space requirements
    - The basic idea here is to pseudo-randomly generate group elements in the form $\alpha^i \cdot \beta^i$, where for every element we keep track of the elements i and j and continue until we obtain a collision of two elements such that

$$\alpha^{i_1} \cdot \beta^{i_1} = \alpha^{i_2} \cdot \beta^{i_2}$$

    - If we substitute $\beta = \alpha^x$ and compare the exponents on both sides of the equation, the collision will lead us to the relation

$$i_1 + xj_1 \equiv i_2 + xj_2 \bmod |G|$$

    - From here, we can compute the discrete logarithm easily using

$$x \equiv \frac{i_2 - i_1}{j_1 - j_2} \bmod |G|$$

    - Pollard's rho method is of great practical importance because it is currently the best known algorithm for computing discrete logarithms in elliptic curve groups

- *Non-generic Algorithms: The Index-Calculus Method*

    - The algorithms defined thus far work for discrete logarithms defined over any cyclic group
    - The index-calculus method does not work for any cyclic group, but it is a very efficient algorithm for computing discrete logarithms in the cyclic groups $\mathbb{Z}_p^* \text{ and } GF(2^m)^*$
    - The index-calculus method relies on the property that a significant factor of elements in G can be efficiently expressed as products of elements of a small subset of G
    - For the group $\mathbb{Z}_p^*$, this means that many elements should be expressible as a product of small primes
    - In order to provide 80 bits of security for DLP based cryptography in the group $\mathbb{Z}_p^*$, the prime p must be at 1024 bits long

8.4: Security of the Diffie-Hellman Key Exchange

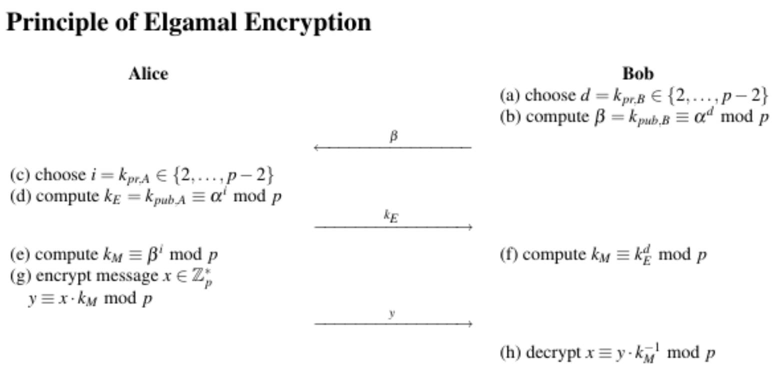
- Let us consider an attacker on the DHKE protocol who can listen to, but not alter, messages
- His goal here, would be to compute the session key k_{AB} shared by two users
- The attacker clearly knows the public parameters α and p
- Additionally, by eavesdropping on the channel during the key exchange, the attacker can obtain the values $A = k_{pub,A}$ and $B = k_{pub,B}$
- So, the question is whether the attacker can compute $k = \alpha^{ab}$ from $\alpha, p, A \equiv \alpha^a \bmod p \text{ and } B \equiv \alpha^b \bmod p$
- This problem is called the *Diffie-Hellman Problem*, and can be generalized to the following
- **Definition 8.4.1:** *Generalized Diffie-Hellman Problem*
    - Given a finite cyclic group G of order n, a primitive element $\alpha \in G$ and two elements $A = \alpha^a$ and $B = \alpha^b$ in G, the Diffie-Hellman problem is to find the group element α^{ab}
- We suppose the attacker knows an efficient method for computing discrete logarithms in $\mathbb{Z}_p^*$, and can thus solve the Diffie-Hellman problem and obtain the key k_{AB} using the following steps

1. Compute user A's private key $a = k_{pr,A}$ by solving the discrete logarithm problem $a \equiv log_{\alpha} A \bmod p$
2. Compute the session key $k_{AB} \equiv B^a \bmod p$

- From the earlier section on DLP attacks, we know that solving the discrete logarithm problem here is infeasible if p is sufficiently large

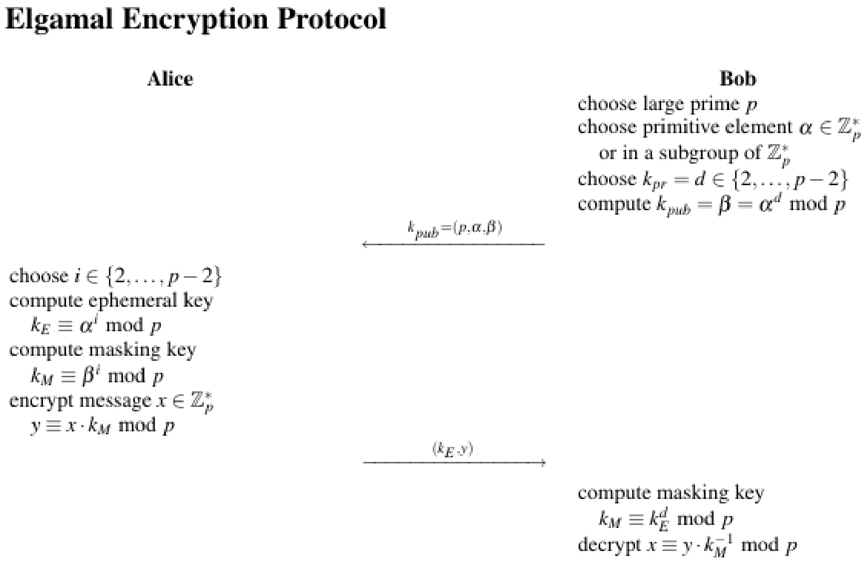
8.5: Elgamal Encryption Scheme

- The *Elgamal* encryption scheme, proposed by Taher Elgamal in 1985, often referred to as *Elgamal encryption*, can be viewed as an extension of the DHKE protocol
- Its security is also based on the intracability of the discrete logarithm problem and the Diffie-Hellman problem
- In this section, we will consider the Elgamal encryption scheme over the group $\mathbb{Z}_p^*$
- **8.5.1: From Diffie-Hellman Key Exchange to Elgamal Encryption**
  - Let us consider two parties, person A and person B
  - Person A wants to send an encrypted message, x, to person B
  - First, both parties perform the Diffie-Hellman key exchange to derived a shared key k_M
    - We can assume the public parameters p and α have been generated
  - Now, person A will used this shared key as a multiplicative mask with which to encrypt x as $y \equiv x \cdot k_M \bmod p$
  - The process is as follows:



- Here, if the key k_M is randomly drawn from $\mathbb{Z}_P^*$, every cipher text $y \in \{1, 2, 3, ..., p-1\}$ is equally likely

- **8.5.2: The Elgamal Protocol**



  - This protocol serves to rearrange the steps from the Diffie-Hellman protocol to where person A needs to send only one message to person B as opposed to two messages

- **8.5.3: Computational Aspects of The Elgamal Protocol**
  - *Key Generation*
    - During the key generation phase, the receiver will generate prime p and then compute both the public and private key
    - Since the Elgamal protocol's security relies on the discrete logarithm problem, p needs to have length of at least 1024 bits
  - *Encryption*
    - The square-and-multiply algorithm will be used to speed up encryption using exponentiation of large numbers