Applied Cryptography and Computer Security

CSE 664 Spring 2017

Lecture 15: Discrete Logarithms and ElGamal Encryption

Department of Computer Science and Engineering
University at Buffalo

What's Next

- **So far we looked at a public-key encryption scheme modulo a composite**
  - **the difficulty of breaking it lies in factoring and computing roots modulo a composite**
- **Now we are going to study a public-key encryption scheme modulo a prime**
  - **discrete logarithms**
  - **Diffie-Hellman problem**
  - **ElGamal encryption**

Terminology Recap

- Recall that **group** G is a set of elements together with
  - a binary operation for which the associative law holds and the set is closed under that operation
  - a unique identity element
  - and unique inverses for each element a of G
- The **multiplicative group modulo** m is denoted by $\mathbb{Z}_m^*$
- A **cyclic group** is one that contains an element a whose powers a^i and a^{-i} make up the entire group
- An element a with such property is called a **generator** of the group

ElGamal Encryption

- **The idea behind ElGamal encryption**
  - **we are given a cyclic multiplicative group G**
  - **let $m \in G$ be an arbitrary element**
  - **if g is an element of G chosen uniformly at random, then so is $g' = g \cdot m$**
    - **m is perfectly protected**
  - **we want g to be pseudorandom**
    - **g is computable using the private key sk and can't be guessed otherwise**

Discrete Logarithm Problem

- **Discrete logarithms**
  - we are given a cyclic group G of order q
  - then there exists an element $g \in G$ such that $G = \langle g \rangle = \{g^i : 0 \leq i \leq q-1\}$
  - for each $h \in G$ there is a unique x such that $g^x = h$
  - such x is called the discrete logarithm of h with respect to g and we use $x = \log_g h$
  - many properties of regular logarithms apply
    - $\log_g 1 = 0$
    - $\log_g(h_1 \cdot h_2) = (\log_g h_1 + \log_g h_2) \bmod q$

Discrete Logarithm Problem

- **The discrete logarithm problem**
  - **in a cyclic group G with given generator g, compute unique $\log_g h$ for a random element $h \in G$**
- **Let PPT algorithm $Set(1^k)$ output a cyclic group G of order q and generator $g \in G$**
- **The discrete logarithm experiment $\mathsf{DLog}_{\mathcal{A},Set}(k)$:**
  1. **Run $(G, q, g) \leftarrow Set(1^k)$ and choose random $h \in G$**
  2. **$\mathcal{A}$ is given G, q, g, h and outputs $x \in \mathbb{Z}_q$**
  3. **the experiment outputs 1 if $g^x = h$ and 0 otherwise**

Discrete Logarithm Problem

- **We say that the discrete logarithm problem is hard (relative to Set) if any PPT adversary $\mathcal{A}$ cannot win the discrete logarithm experiment with a non-negligible probability**

$$\mathsf{Pr}[\mathsf{DLog}_{\mathcal{A},Set}(k) = 1] \leq \mathsf{negl}(k)$$

- **When is the discrete logarithm problem hard?**
  - **most often a multiplicative group modulo prime p, $\mathbb{Z}_p^*$, or its subgroup is used**
  - **the choice of parameters is driven by known algorithms for solving the discrete logarithm problem**

Discrete Logarithm Problem

- **Is the discrete logarithm problem generally hard?**
  - **it is hard in** $(\mathbb{Z}_p^*, \cdot)$ **using proper parameters**
  - **how about** $(\mathbb{Z}_n, +)$**?**
    - **let** $gcd(g, n) = 1$**, so** g **is a generator of** $\mathbb{Z}_n$
    - **now** $g^x \bmod n$ **in multiplicative groups translates to** $gx \bmod n$
    - **the discrete logarithm problem is then** $gx \equiv h \pmod{n}$
    - **but since** $gcd(g, n) = 1$**, we can compute** $g^{-1} \bmod n$
    - **now** $x = \log_g h = hg^{-1} \bmod n$
  - **are there other cases when** $\log_g h$ **is hard?**

Discrete Logarithm Problem

- Algorithms for solving discrete log
  - there are generic algorithms that work for every cyclic group
    - e.g., Shanks' method, Pollard rho method, Pohlig-Hellman algorithm
  - there are algorithms that work for certain groups only
    - they rely on particular representation of the group
    - they are faster and will require larger security parameters
    - e.g., general number field sieve for $(\mathbb{Z}_p^*, \cdot)$ with prime p
  - for certain groups, there are no better attacks than generic algorithms
    - e.g., groups over elliptic curves

Algorithms for Discrete Logarithm Problem

- **Shanks' baby-step/giant-step algorithm**
  - **the algorithm requires $O(\sqrt{q})$ steps for groups of order q**
  - **"giant steps" are of size $\sqrt{q}$ and "baby steps" are of size 1**
  - **algorithm steps on input $G = \langle g \rangle$ and $h \in G$**
    1. **set $t = \lfloor \sqrt{q} \rfloor$**
    2. **for $i = 0$ to $\lfloor q/t \rfloor$, compute $g_i = g^{i \cdot t}$**
    3. **sort pairs (i, g_i) by second value**
    4. **for $i = 0$ to t, compute $h_i = h \cdot h^i$; if $h_i = g_j$ for some j, return $jt - i \bmod q$**
  - **taking into account sorting and mod exponentiations & multiplications, overall complexity is $O(\sqrt{q} \cdot \mathsf{polylog}(q))$**

Algorithms for Discrete Logarithm Problem

- **Pohlig-Hellman algorithm**
  - **works when factorization of group order q is known or can be computed**
  - **reduces the problem of computing discrete log in groups of order $q = q_1 \cdot q_2$ to discrete log in groups of order q_1 and q_2**
  - **it uses a variant of the Chinese remainder theorem**
  - **thus, group order q must always contain at least one large factor**

Discrete Logarithm Problem

- **Discrete logarithm problem**
  - **groups of prime order are a popular choice because..**
    - **Pohlig-Hellman algorithm is not effective**
    - **finding a generator is easy: each element is a generator**
    - **exponent manipulation is easier: each exponent has an inverse**
    - **other security assumptions are more likely to hold**
- **How do we produce a group of prime order or with a large factor in the group order?**

Discrete Logarithm Problem

- **A subgroup of a group is a subset of the group that forms a group with the same binary operation**
  - **for example, powers a^i in a multiplicative group can "hit" only a subset of the group elements**
- **A group $(\mathbb{Z}_p^*, \cdot)$ for prime p has order $\phi(p) = p - 1$**
- **Subgroups of $\mathbb{Z}_p^*$ can have any order q that divides $p - 1$**
  - **we often might want $p = 2q + 1$ for prime p and q**
  - **or we can have $p = 2qt + 1$ for reasonably large prime q and some t**

Discrete Logarithm Problem

- **How do we generate a subgroup?**
  - **let order of some group G be $q = q_1 \cdot q_2$**
  - **to obtain generator of subgroup of order q_1, we often can pick $a \in G$ and set $g = a^{q_2}$**
- **How do we generate proper prime p?**
  - **we want to generate primes p and q such that $q | (p - 1)$**
  - **to do so, we need to know the factorization of $p - 1$**
  - **approach 1: generate a random prime p and then factor $p - 1$**
  - **approach 2: generate a random q first and then choose r such that $p = 2rq + 1$ is prime**

Discrete Logarithm Problem

- **The choice of parameters**
  - **for group $\mathbb{Z}_p^*$, p needs to be large enough for discrete log to be hard to solve**
    - **fastest algorithm runs on average in $2^{O(k^{1/3}(\log k)^{2/3})}$ time**
    - **today this requires $|p|$ of 1536 bits or higher**
  - **the group order must have at least one large factor to prevent exhaustive search**
    - **e.g., 192 bits or higher**
  - **if the group order is prime, it can be relatively short (same 192 bits)**
    - **this can improve performance**

ElGamal Public-Key Encryption

- **ElGamal encryption**
  - **a public-key encryption published in 1985 by ElGamal**
  - **its security relies on the discrete logarithm problem being hard**
  - **the encryption operation is randomized**
  - **a ciphertext is twice as long as the original message**
  - **the idea:**
    - **the plaintext m is masked by multiplying it by h^y**
    - **another value g^y is transmitted as part of the ciphertext**
    - **knowing the private key, one can compute h^y from g^y and unmask the message**

ElGamal Public-Key Encryption

- **Key generation**
  - **choose a cyclic group** G **of order** q **and a generator** $g \in G$
  - **choose a random** x **from** $\mathbb{Z}_q$ **and compute** $h = g^x$
  - **public key:** $pk = (G, q, g, h)$
  - **private key:** $sk = x$
- **Encryption**
  - **to encrypt a message** $m \in G$ **using public key** $pk = (G, q, g, h)$
  - **choose a random number** $y \in \mathbb{Z}_q$
  - **compute the ciphertext as**
    $c = \mathsf{Enc}_{pk}(m) = (c_1, c_2) = (g^y, m \cdot h^y)$

ElGamal Public-Key Encryption

- **Decryption**
  - **given a ciphertext** $c = (c_1, c_2)$ **and keys** $pk = (G, q, g, h)$**,** $sk = x$
  - **decrypt the ciphertext as** $m = \mathsf{Dec}_{sk}(c) = c_2 \cdot c_1^{-x}$
- **Correctness**
  - **we show that** $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$
  - **the decryption is**

ElGamal Public-Key Encryption

- **Example**
  - **key generation**
    - **let** $p = 2579$ **and** $g = 2$
    - **suppose we choose** $x = 765$ **and compute** $h = 2^{765} \bmod 2579 = 949$
    - **the public key is** $pk = (2579, 2, 949)$
    - **the secret key is** $sk = 765$
  - **encryption**
    - **to encrypt** $m = 1299$**, suppose we choose** $y = 853$
    - **to encrypt, first compute** $c_1 = 2^{853} \bmod 2579 = 435$ **and** $c_2 = 1299 \cdot 949^{853} \bmod 2579 = 2396$

ElGamal Public-Key Encryption

- **Example (cont.)**
  - **encryption**
    - **the ciphertext is** $c = (435, 2396)$
  - **decryption**
    - **given** $c = (c_1, c_2)$**, we decrypt the message by computing** $c_2 \cdot (c_1^x)^{-1} \bmod p$**:**

$$m = 2396 \cdot (435^{765})^{-1} \bmod 2579 = 1299$$

- **Security of ElGamal encryption**
  - **depends on the discrete logarithm problem in** G **being infeasible**
  - **but it is based on a different hardness assumption**

ElGamal Public-Key Encryption

- **On parameter choice in ElGamal**
  - **if g is a generator of $\mathbb{Z}_p^*$**
    - **p should have several hundred digits for the discrete logarithm to be hard**
    - **$p-1$ should have at least one large prime factor**
  - **if g does not generate $\mathbb{Z}_p^*$**
    - **the order of g can be smaller than $p-1$**
    - **for instance, prime order q of length 192 bits is sufficient**
  - **the group doesn't have to be $\mathbb{Z}_p^*$**
    - **other choices include groups defined over elliptic curves**

ElGamal Public-Key Encryption

- **On parameter choice in ElGamal**
  - **sharing parameters**
    - **unlike in cryptosystems that use composite modulus, here the same p and g can be used in many keys**
    - **when modulus $n = pq$ is used, its factorization is often the private key or allows to compute the private key**
    - **here both p and g are public**
    - **that makes this encryption secure is the knowledge of the secret value x**
  - **a fresh value of y should be picked for each encryption**

Diffie-Hellman Key Exchange

- **Diffie-Hellman key exchange protocol**
  - **Alice and Bob want to compute a shared key, which must be unknown to eavesdroppers**
  - **Alice and Bob share public parameters: a group G of order q and a generator g**
  - **Alice randomly chooses $x \in \mathbb{Z}_q$ and sends g^x to Bob:** $A \xrightarrow{g^x} B$
  - **Bob randomly chooses $y \in \mathbb{Z}_q$ and sends g^y to Alice:** $A \xleftarrow{g^y} B$
  - **the shared secret is set to g^{xy}**
    - **Alice computes it as $(g^y)^x = g^{xy}$**
    - **Bob computes it as $(g^x)^y = g^{xy}$**

Diffie-Hellman Key Exchange

- **Diffie-Hellman key exchange protocol**
  - **Alice and Bob are able to establish a shared secret with no prior relationship**
  - **it is believed to be infeasible for an eavesdropper to compute g^{xy} given g^x and g^y**
- **Diffie-Hellman problem**
  - **Computational Diffie-Hellman (CDH) problem**
    - **given g, g^x and g^y, compute g^{xy}**
  - **Decision Diffie-Hellman (DDH) problem**
    - **given g, g^x, g^y, and g^z, determine whether $xy = z$ (modulo q)**

Diffie-Hellman Problem

- **As before, CDH (DDH) problem is hard if any PPT adversary $\mathcal{A}$ has at most negligible probability in solving it**

- **Diffie-Hellman problem**
  - **DDH is a stronger assumption than CDH**
    - **breaking CDH implies breaking DDH, but the converse is not true**
  - **discrete log is at least as hard as CDH**
  - **security of the Diffie-Hellman key exchange protocol is based on the CDH assumption**

Security of ElGamal Encryption

- **Going back to ElGamal**
  - **if the CDH assumption holds, ElGamal is one-way**
    - **i.e., if you can solve the CDH problem, you will be able to decrypt**
  - **if the DDH assumption holds, ElGamal is secure in the sense of indistinguishability**

- **Formally: if the DDH problem is hard (relative to Set), then the ElGamal encryption scheme has indistinguishable encryptions under a chosen-plaintext attack**

Security of ElGamal Encryption

- **Security proof sketch**

Security of ElGamal Encryption

- **Care must be taken when mapping messages to group elements**
  - **one (least significant) bit of discrete logarithm is easy to compute**
  - **we say that an element $y \in \mathbb{Z}_p^*$ is a quadratic residue (QR) modulo p if there exists $x \in \mathbb{Z}_p^*$ such that $y = x^2 \bmod p$**
  - **given a ciphertext, an adversary can tell whether the underlying plaintext was a QR modulo p or not**
  - **to ensure indistinguishability, we need to make sure that all values we use will have the same value for that bit**
  - **thus, we encode messages as $x^2 \bmod p$ only**

ElGamal Encryption

- **More on quadratic residues**
  - a **is called a quadratic residue (QR) modulo** m **if** $x^2 \equiv a \pmod{m}$ **has solutions**
  - a **is called a quadratic non-residue (QNR) otherwise**
  - **there are the same number,** $(p-1)/2$**, of QRs and QNRs modulo prime** p **among numbers** $1, \ldots, p-1$
  - **if** a **is a QR modulo prime** p**, its square roots are** $\pm x \bmod p$
    - **then one solution is** $\leq (p-1)/2$ **and the other one is not**
  - **square roots are efficiently computable**

ElGamal Encryption

- **Encryption with ElGamal becomes**
  - **given a message m, interpret it as a integer between 1 and q, where $q = (p-1)/2$**
  - **compute $\hat{m} = m^2 \bmod p$ and encrypt $\hat{m}$**
  - **upon decryption:**
    - **obtain $\hat{m}$**
    - **compute square roots m_1, m_2 of $\hat{m}$ modulo p**
    - **set m to the unique $1 \leq m_i \leq q$**

Summary

- **The discrete logarithm problem is considered hard in groups modulo a large prime**
- **Many constructions rely on it**
- **ElGamal is an example of encryption that assumes hardness of discrete logarithm problem**
- **Diffie-Hellman key exchange is built using similar assumptions**
  - **two types of hardness assumptions are known as computational and decision Diffie-Hellman problems**
- **ElGamal is CPA-secure under the DDH assumption**