Introduction to Cryptography
CS 355
Lecture 24

Diffie-Hellman and Discrete Log
Lecture Outline

- The Discrete Log problem
- The Diffie-Hellman protocol
Discrete Logarithm Problem (DLP)

- Given a multiplicative group \((G, \ast)\), an element \(g\) in \(G\) having order \(n\) and an element \(y\) in the subgroup generated by \(g\), denoted \(<g>\)
- Find the unique integer \(x\) such that
  \[ g^x \mod n = y \]
- i.e., \(x\) is the discrete logarithm \(\log_g y\)
- For example, given the group \(\mathbb{Z}_p^*\), where \(p\) is a 1024-bit prime, let \(g\) be an element having order \(q\), where \(q\) is a 160-bit prime
  - \(q \mid (p-1)\)
  - e.g., \(\mathbb{Z}_7^* = \{3, 2, 6, 4, 5, 1\}\), we choose the subgroup \(\{2, 4, 1\}\)
The Diffie-Hellman Protocol

- Key agreement protocol, both A and B contribute to the key
- Setup: p prime and g generator of $\mathbb{Z}_p^*$, p and g public.

$$K = (g^b \mod n)^a = g^{ab} \mod p$$

Pick random, secret a
Compute and send $g^a \mod p$

K = (g^a \mod n)^b = g^{ab} \mod p

Pick random, secret b
Compute and send $g^b \mod p$
Diffie-Hellman Key Establishment

• A and B wishes to establish a shared secret key so that no eavesdropper can compute the key:
• A and B shares public parameters a group $Z_p$ and a generator $g$
  – A randomly chooses $x$ and send $g^x \mod p$ to B
  – B randomly chooses $y$ and send $g^y \mod p$ to A
  – Both A and B can compute $g^{xy} \mod p$
  – It is (believed to be) infeasible for an eavesdropper to compute $g^{xy} \mod p$
  – A and B can establish a shared secret without sharing any secret to start with
CDH and DDH

• Security of the Diffie-Hellman key establishment protocol based on the CDH problem

• Computational Diffie-Hellman (CDH)
  – Given a multiplicative group \((G, \ast)\), an element \(g \in G\) having order \(q\), given \(g^x\) and \(g^y\), find \(g^{xy}\)

• Decision Diffie-Hellman (DDH)
  – Given a multiplicative group \((G, \ast)\), an element \(g \in G\) having order \(q\), given \(g^x, g^y,\) and \(g^z\), determine if \(g^{xy} \equiv g^z \mod n\)

• Discrete Log is at least as hard as CDH, which is at least as hard as DDH.
Choices of Parameters

• Why use an element of order $q$, instead of just using a generator for $\mathbb{Z}_p^*$?

• Answer:
  – it is often beneficial to have order being a prime
    • e.g., given $e$, one can find $d$ s.t. $g^{ed} = g$
  – Balance security and size
    • $p$ needs to be large enough for discrete log to be hard, thus 1024 bits
    • we want the group to be relative small, so that an index to an element in the group is short (e.g., 160 bits)
      – it needs to be large enough to prevent exhaustive search
Algorithms for The Discrete Log Problem

• There are generic algorithms that work for every cyclic group
  – Pollard Rho
  – Pohlig-Hellman

• There are algorithms that work just for some groups such as $\mathbb{Z}_p^*$
  – e.g., the index calculus algorithms
  – these algorithms are much more efficient
  – therefore, 1024 bits are needed for adequate level of security
Bit Security in Discrete Log

- Even though it is difficult to find $\log_g x$, it is possible to determine some bits in $\log_g x$
  - e.g., let $g$ be the generator of $\mathbb{Z}_p^*$, consider the least significant bit (LSB) of $\log_g x$
    - recall that $\log_g x$ is even iff. $x$ is quadratic residue in $\mathbb{Z}_p^*$
  - However, finding some bits (aka. hard-core bits) is as hard as computing discrete log
    - in $\mathbb{Z}_p^*$, when $p-1=2^st$, where $t$ is odd, computing the $s$ least significant bits are easy, computing the $s+1$ LSB is difficult
One Way Functions

• A function $f(x)$ is a one-way function if
  – given $a$, it is easy to compute $f(a)$.
  – yet given $b$, it is difficult to find $a$ such that $f(a)=b$.

• Examples of one-way functions
  – Modular exponentiation $f(x) = g^x \mod p$
  – Multiplication $f(x,y) = x \cdot y$

• One way functions are the foundations for modern cryptography, yet we do not know whether they exist or not.
  – existence of one-way functions imply $P \neq NP$
Coming Attractions …

- ElGamal Encryption