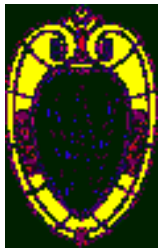

CS/ECE/Math 435
Introduction to Cryptography

Professor Chris DeMarco
Department of Electrical & Computer Engineering
University of Wisconsin-Madison

Spring Semester 2021

Concepts underlying Diffie-Hellman prove useful in a number of other security constructs, beyond encryption (e.g., authentication & digital signatures).

Critical D-H concepts are:

- One first selects (large) prime number p, and identifies special $g \in \mathbb{Z}_p^*$ having property that g is "generator" (synonomously, a "primitive root") of $\mathbb{Z}_p^*$.

- For such a g, the exponentiation function $\mod(g^x, p)$ is a one-to-one, onto mapping of $\mathbb{Z}_p^*$ to $\mathbb{Z}_p^*$. And importantly, computational cost of evaluating the exponentiation function is relatively low (Bach notes section 3.4 show that for $x < p$, number of multiplies require to compute is $\leq 2 \cdot \log_2(p)$)

- Given $a = \mathrm{mod}(g^x, p)$, computing x from a (the inverse of the exponentiation function) is known as the discrete logarithm.

Importantly, the best known algorithms for discrete logarithm have computational cost much higher than exponentiation (order "big O" of $\sqrt{p}$, $O(\sqrt{p})$).

RSA encryption, similar to D-H key exchange, exploits this asymmetry: a function "easy" to compute in forward direction, "hard" to invert.

First, examine a variant on D-H key exchange, a U.S. government published standard known as KEA (creatively, Key Exchange Algorithm).

Initialization of KEA: • Select integer q prime, $q < 2^{160}$, so q has 160-bit binary representation.

• Select integer p prime, p having 1024-bit representation.

• Identify g that is a generator for Z_q^*. Observe that computational cost for an attacker seeking to compute the discrete logarithm will be order $\sqrt{q} = 2^{80} \cong 1.2 \times 10^{24}$

- q, p and g are made publically available by the network protocol manager.

- User A selects a first internally secret key X_A, and publishes a public key of g^{X_A} to all other users.

- Similarly, user B selects x_B and publishes g^{x_B}.

- For a particular communication session between A and B,

  A selects random r_A, internally secret.

  B " " r_B, " "

  A sends g^{r_A} to B

  B sends g^{r_B} to A

• Shared key w computed as:

A: $\underbrace{\text{mod}\left\{\left[g^{x_B}\right]^{r_A} + \left[g^{r_B}\right]^{x_A}, p\right\}}$

$\Big\Vert$

w

$\Big\Vert$

B: $\text{mod}\left\{\left[g^{x_A}\right]^{r_B} + \left[g^{r_A}\right]^{x_B}, p\right\}$

New Applications

All of our development of "tools" in 435 have focused on the encryption problem: enable confidential message/file exchange between authorized readers, with unauthorized entities (attacker) facing very high probability of very high computational cost to decrypt message.

These same tools can be adapted to a set of closely related communication security problems

- Authenticity: confirming identity of the source of a message;
- Integrity: confirming that message as received is unaltered from message as sent;

- Commitment: (closely related to authenticity, but not identical):
  once a user creates a message or file, ensure that they are unable to repudiate, i.e. unable to falsely claim that it was created or altered by other entities.

Most common authentication problem in application is that of password-based access, "logging in."

Assume: A centralized entity (e.g. network manager) must be able to evaluate a user's request for access, based on password information.

But: Password itself should not be:

i) stored by central entity

ii) communicated openly on public network

D-H like concepts provide easy answer to (i) & (ii): a "one-way" function, $e(x)$, for which

$$e(x) = \mathrm{mod}(g^x, p)$$ is a common choice.

Ideas:

$y = e(x)$ should be low-cost computation;

$x = e^{-1}(y)$ should be high-cost computation.

The actual password plays role of the message x;
x itself is never publically communicated, nor known to any entity other than its owner.
Only $e(x)$ is commicated/stored.

Of course, if access is granted based on match of $y = e(x)$, one may desire further encryption even in communicating y.

Common step to improve password security: "salting" - parameterize password encryption algorithm with an additional, randomly generated key parameter, S.

Again, for historical reasons, a parallel terminology. Instead of terming this a "key," in this context "S" is known as the "salt."

- The one-way function may now take two arguments

$$e(x, s)$$

x: actual password

s: the "salt," or key

- Improvements possible, at cost of imposing some "low-cost" (but not entirely trivial) computation on the user requesting access.

RSA-based password/authentication protocol.

- User requesting access: entity "A"
  Centralized entity managing access: "B"

- Consistent with RSA system, A chooses primes p, q, with $N = p \cdot q$, and $\varphi = (p-1) \cdot (q-1)$. A further chooses integers d, e such that $\text{mod}(d \cdot e, \varphi) = 1$.

- (N, e) are public; d, p, q remain internally secret to A. A's identity tied to A publishing (N, e).

  Recall, based on above, RSA defines

  Encryption function $E(x) = \text{mod}(x^e, N)$
  (uses public information)

  Decryption function: $D(y) = \text{mod}(y^d, N)$
  (uses information internally secret to A)

- User A sends access request to B;
- B generates a randomly selected message "y", sends y to A.
- A computes $x = D(y)$, sends x back to B
- B tests $y \stackrel{?}{=} E(x)$

(yes $\Rightarrow$ A confirmed associated to (N, e))

Alternatives / Supplements to password access : "keyed" client-server interactions.

In modern networks, many communication processes benefit from a secure protocol, but the volume and frequency of these communications make user intervention (e.g., distinct password entry) impractical.

A commonly used approach employs time-stamped, limited duration keys.

Kerberos Protocol

- Communication session desired between users U and V, facilitated by "trusted" key generator entity T.

- U and V must have secure communication with T; typically this *is* established by password authorized access (again; motivation is to enable *many* $U-V$ interactions "on-the-fly"; we can afford more thorough, password authentication of $U-T$, $V-T$ when joining network)

- A critical element: each communication will have a beginning timestamp, "t", and a duration or "lifetime", "l".