Diffie-Hellman key-exchange protocol

This protocol allows two users to choose a common secret key, for DES or AES, say, while communicating over an insecure channel (with eavesdroppers).

The two users agree on a common large prime p and a constant value a, which may be publicly known and available to everyone. It is best if the smallest exponent $e > 0$ for which $a^e \equiv 1 \pmod{p}$ is $e = p - 1$, but the protocol will work if $e < p - 1$ provided e is still large. When $e = p - 1$, a is called a primitive root modulo p. In that case the numbers

$$a^1 \bmod p,\ a^2 \bmod p,\ a^3 \bmod p,\ \ldots,\ a^{p-1} \equiv 1 \bmod p$$

are all different and form an RSR modulo p.

Alice secretly chooses a random x_A in $0 < x_A < p-1$ and computes $y_A = a^{x_A} \bmod p$. Bob secretly chooses a random x_B in $0 < x_B < p-1$ and computes $y_B = a^{x_B} \bmod p$.

Alice sends y_A to Bob. Bob sends y_B to Alice. An eavesdropper, knowing p and a, and seeing y_A and y_B, cannot compute x_A or x_B from this data unless he can solve the Discrete Logarithm Problem quickly.

Alice computes $K_A = y_B^{x_A} \bmod p$.

Bob computes $K_B = y_A^{x_B} \bmod p$.

Then

$$K_A \equiv a^{x_A \cdot x_B} \equiv K_B \pmod{p}$$

and $0 < K_A, K_B < p$, so $K_A = K_B$.

Alice and Bob choose certain agreed-upon bits from K_A to use as their key for a single-key cipher like DES or AES.

Although this protocol provides secure communication between Alice and whoever is at the other end of the communication line, it does not prove that Bob is the other party. To guarantee that Bob is at the other end, they would have to use a signature system like RSA.

Discrete Logarithms

The Diffie-Hellman key exchange and several other crypto algorithms could all be broken if we could compute discrete logarithms quickly, that is, if we could easily solve the exponential congruence $a^x \equiv b \bmod p$.

By analogy to ordinary logarithms, we may write $x = \log_a b$ when p is understood from the context. These discrete logarithms enjoy many properties of ordinary logarithms, such as $\log_a bc = \log_a b + \log_a c$, except that the arithmetic with logarithms must be done modulo $p - 1$ because $a^{p-1} \equiv 1 \bmod p$.

Neglecting powers of $\log p$, the congruence may be solved in $\mathrm{O}(p)$ time and $\mathrm{O}(1)$ space by raising a to successive powers modulo p and comparing each with b. It may also be solved in $\mathrm{O}(1)$ time and $\mathrm{O}(p)$ space by looking up x in a precomputed table of pairs $(x, a^x \bmod p)$ sorted by the second coordinate.

The RSA public-key cipher

Rivest-Shamir-Adleman. Let $n = pq$ be the product of two large primes.
Then $\phi(n) = \phi(pq) = (p-1)(q-1)$, so
$ed \equiv 1 \pmod{(p-1)(q-1)}$.

Encode plaintext as (blocks) $0 \le M < n$.
Encipher M as $C = E(M) = M^e$ mod n.
Decipher C as $M = D(C) = C^d$ mod n.

This works, that is, $D(E(M)) = M$ for all M in $0 \le M < n$, provided that $ed \equiv 1 \pmod{\phi(n)}$ since $M^{\phi(n)} \equiv 1 \pmod{n}$ by Euler's Theorem. This is true since $\phi(n) = (p-1)(q-1)$ and $ed \equiv 1 \pmod{(p-1)(q-1)}$.

This implies that e and d must each be relatively prime to $\phi(n)$.

Each user of RSA has her own set of keys: Make n and e public, but keep d secret. The factors p and q are not needed after e and d are computed, but in any case should not be revealed.

If many users wish to communicate securely in pairs, then RSA requires fewer total keys to be stored than Pohlig-Hellman.

Cryptanalysis: Since n is public and one can easily compute d from e and the factors of n, a direct approach to breaking RSA is to factor n. Using the best currently-known methods, this is about as hard as solving the Discrete Logarithm Problem with the same sized modulus. For a modulus n of 400 decimal digits, this is too hard for current algorithms and computers.

Exponentiation ciphers

RSA is an example of an exponentiation cipher, that is, a cipher in which encryption and decryption are done by raising the plain or cipher text to a secret power modulo a large integer.

Suppose the modulus is a large integer n. Encode plaintext as (blocks) $0 \le M < n$. Encipher M as $C = E(M) = M^e \bmod n$. Decipher C as $M = D(C) = C^d \bmod n$.

This works, that is, $D(E(M)) = M$ for all M in $0 \le M < n$, provided that $ed \equiv 1 \pmod{\phi(n)}$ since $M^{\phi(n)} \equiv 1 \pmod{n}$ by Euler's Theorem.

Proof: Write $ed = t\phi(n) + 1$ for some integer t.

This implies that e is relatively prime to $\phi(n)$.

In the case of RSA, n is the product of two primes so large that n cannot be factored in a reasonable time.

Pohlig-Hellman cipher

This is another exponentiation cipher.

This is NOT a public-key cipher.

Let $n = p =$ prime. Then $\phi(p) = p - 1$ and $ed \equiv 1 \pmod{p-1}$.

Method 1: Keep all of p, e, d secret. All three are the “key”. There is just one user or one pair of users.

Method 2: Let p be public and keep e and d secret. The key is the pair (e, d). Each user has a secret pair to safeguard her personal secrets. Each pair of users who wish to communicate choose a key pair.

Since it may take a while to generate a large prime, Method 2 is more common than Method 1. Furthermore, Method 2 has interesting mathematical properties which foster its use in special ways discussed later (Massey-Omura, mental poker).

Cryptanalysis: For a known-plaintext attack on Method 2, one is given a prime p, C and M, and must find an exponent e so that $C \equiv M^e \pmod{p}$ or (equivalently) d so that $M \equiv C^d \pmod{p}$, that is, the attacker must solve a Discrete Logarithm Problem. Although there are some easy cases, such as $m = p =$ prime where $p - 1$ has only small prime factors, the general case is about as difficult to solve as it is to factor a general number as large as m.

An important property of the Pohlig-Hellman cipher

Let p be a large prime. Suppose users A and B have encryption algorithms E_A and E_B and decryption algorithms D_A and D_B. (So $E_A(M) = M^{e_A} \bmod p$, $D_A(C) = C^{d_A} \bmod p$, where $e_A d_A \equiv 1 \pmod{p-1}$, etc.) Since the encryption and decryption algorithms are all exponentiation modulo a fixed modulus, they all *commute*, that is, they may be done in any order and give the same result. For example, $E_A(D_B(x)) = D_B(E_A(x))$ for every x because both are just $x^{e_A d_B} \equiv x^{d_B e_A} \bmod p$.

RSA Signatures

RSA has no direct authentication: Anyone can send any message to you and claim it came from anyone. However, one can sign RSA messages as follows:

Use the same notation for enciphering and deciphering algorithms as we did for Pohlig-Hellman: E_A, D_B, etc. Alice can sign (and encipher) a message M to Bob by sending $C = E_B(D_A(M))$ to Bob. Bob can decipher C by applying D_B to it (to get $D_A(M)$) and then check the signature by applying E_A to the latter.

Note that Bob's cipher algorithms do not commute with Alice's because the modulus is different. Thus the order in which Bob applies the operations to C matters: Bob must do D_B first and then E_A second.

There is another problem caused by the different moduli. D_A and E_A do arithmetic modulo Alice's modulus n_A while E_B and D_B do arithmetic modulo Bob's modulus n_B. This works fine if $n_A < n_B$ but part of the message will be lost if $n_A > n_B$. There are three ways to solve this problem: 1. Re-block the message after D_A is applied. 2. Enforce an arbitrary threshold T and let every RSA user A have two complete sets of RSA keys, one with $n_{A_1} < T$ and one with $n_{A_2} > T$. The keys with the smaller modulus n_{A_1} are used for signing messages from A and the keys with the larger modulus n_{A_2} are used to encipher messages going to A.

3. A more elegant solution is for Alice to sign (and encipher) a message M to Bob by sending $C = E_B(D_A(M))$ to Bob when $n_A < n_B$, and by sending $C = D_A(E_B(M))$ to Bob when $n_A > n_B$. In either case, Bob undoes these operations in reverse order. What if Alice later denies sending M and Bob goes to an independent judge to prove that M bears Alice's signature? In the first case ($n_A < n_B$), Bob gives the judge M and $X = D_B(C)$, the judge computes $M' = E_A(X)$ and tests whether $M' = M$. If so, the judge rules that Alice signed M. In the second case ($n_A > n_B$), Bob gives the judge M and C, the judge computes $X = E_B(M)$ and $X' = E_A(C)$ and tests whether $X' = X$. If so, the judge rules that Alice signed M.

The El-Gamal public-key cipher

The **ElGamal public key cryptosystem** is strictly not an exponentiation cipher, although exponentiation is done during enciphering and deciphering.

Fix a large prime p and a primitive root g modulo p in $1 < g < p$, both of which are public. Each user A who wishes to participate in this public-key cryptosystem chooses a secret a_A in $0 < a_A < p - 1$ and publishes $b_A = g^{a_A} \bmod p$. When a user B wants to send a secret message M in $0 < M < p$ to A, she chooses a random k in $0 < k < p - 1$ and sends to A the pair

$$C = (g^k \bmod p, (Mb_A^k) \mod p).$$

The plaintext M is enciphered by multiplying it by b_A^k in the second component of C. Note that $b_A^k \equiv (g^{a_A})^k \equiv g^{a_A k} \pmod{p}$. The first component of C provides a hint for deciphering M from the second component of C, but one which is useful only to A. Only A knows the secret key a_A, so only A can compute $(g^k)^{a_A} \equiv g^{a_A k} \pmod{p}$. If the multiplicative inverse of this number is multiplied times the second component, one recovers M:

$$\left(g^{a_A k}\right)^{-1}\left(M b_A^k\right) \equiv \left(g^{a_A k}\right)^{-1}\left(M g^{a_A k}\right) \equiv M \ (\mathrm{mod}\ p).$$

Cryptanalysis of the El-Gamal public-key cipher

An eavesdropper who could solve the discrete logarithm problem modulo p could compute M from C and public data without knowing a_A as follows. The first component of C is $h = g^k \bmod p$. This number and $T = (Mb_A^k) \bmod p$ are observed by the eavesdropper. The eavesdropper knows p and g because these numbers are public. He can also obtain A's public key b_A from A's directory, just as B did. He would solve the discrete logarithm problem $g^k \equiv h \pmod{p}$ for k and then compute

$$T\left(b_A^k\right)^{-1} \equiv \left(Mb_A^k\right)\left(b_A^k\right)^{-1} \equiv M \pmod{p}.$$

The Massey-Omura public-key cipher

One can change the Pohlig-Hellman private-key cipher slightly to make a public-key cipher. This was done by Massey and Omura. Their system is not used much because it is inefficient. (But the elliptic curve version is used.)

Consider a Pohlig-Hellman cipher with common prime p. This was called Method 2 earlier. Suppose users A and B have encryption algorithms E_A and E_B and decryption algorithms D_A and D_B. (So $E_A(M) = M^{e_A} \bmod p$, $D_A(C) = C^{d_A} \bmod p$, where $e_A d_A \equiv 1 \pmod{p-1}$, etc.) Since the encryption and decryption algorithms are all exponentiation modulo a fixed modulus, they all *commute*, that is, they may be done in any order and give the same result. For example, $E_A(D_B(x)) = D_B(E_A(x))$ for every x because both are just
$x^{e_A d_B} \equiv x^{d_B e_A} \bmod p$.

How do A and B use this property as a public-key cipher? The "public key" is the common prime modulus p. The private keys are ALL of the exponents (unlike RSA). If Alice wants to send a message $0 < M < p$ to Bob, she first sends $E_A(M)$ to Bob. Bob replies by sending $E_B(E_A(M))$ to Alice. Then Alice sends

$$D_A(E_B(E_A(M))) = E_B(D_A(E_A(M))) = E_B(M)$$

to Bob. Bob deciphers the message by applying D_B to it.

The security depends on the difficulty of the Discrete Logarithm Problem. The system is a protocol which requires a two-way exchange of three messages—not impossible, but still less convenient than RSA or El-Gamal in which just one message is sent.

Mental poker

Each player is dealt five of the 52 cards. Each player can see his hand and not any other player's hand. Players bet based on their hands. The "best" hand wins. In some variations, some cards are revealed and some cards may be replaced by cards not yet dealt.

The "e-mail" or "mental" protocol for poker requires a fair deal: Players see their own hands, but not other hands. The hands are disjoint. All hands are equally likely. A player can "draw" (replace) selected cards. A player can reveal individual cards one at a time without revealing other cards. All players can check at the end of the game that there was no cheating.

We use a variation of Pohlig-Hellman to implement mental poker.

Assume there are two players, Alice and Bob. (There are similar protocols for three or more players.)

The players jointly choose a large prime p as modulus. Each secretly chooses e_A, d_A, e_B, d_B, as in Pohlig-Hellman. Define $E_A(M) = M^{e_A} \bmod p$, etc. Recall that these functions commute: $E_A \circ E_B = E_B \circ E_A$, etc. Let $M_1, \ldots, M_{52}$ be the encoded) deck (more if there is a joker).

1. Bob enciphers the cards as $C_i = E_B(M_i)$ for $i = 1, \ldots, 52$. Bob sorts the C_i and sends them to Alice.

2. Alice selects five cards C_i at random and sends then to Bob, who decrypts them as his hand.

3. Alice selects five more random cards, say $C_1, \ldots, C_5$ (her hand) and enciphers them as $C'_i = E_A(C_i)$. She sends them to Bob.

4. Bob deciphers the C'_i (they are still enciphered with E_A after he applies D_B to undo E_B) and sends them back to Alice.

5. Alice deciphers the five cards and uses them as her hand. They bet and play poker.

6. At the end of the hand, Alice and Bob exchange their keys e_A, etc., and check everything that happened.

Quadratic residues

Unfortunately, one can cheat in mental poker because E_A, E_B, etc., preserve quadratic residues.

Definition. a is a *quadratic residue* modulo n if gcd$(a, n) = 1$ and there is an integer x such that $x^2 \equiv a \pmod{n}$. Such an x called a *square root* of a modulo n. a is a *quadratic non-residue* modulo n if gcd$(a, n) = 1$ and there is no integer x such that $x^2 \equiv a \pmod{n}$.

Theorem. Let $0 < a < n$, gcd$(a, n) = 1$, and gcd$(e, \phi(n)) = 1$. Then a is a QR mod n if and only if a^e is a QR mod n.

Alice can use this theorem to cheat: Perhaps most high cards are QR and most low cards are QNR. It is like playing with a deck in which most high cards are “marked”. This attack can be foiled by (a) appending extra bits to each M_i or (b) multiplying some M_i by a fixed QNR in order to make all cards be QR or all cards be QNR.

Note: When the modulus is prime, we have: QR $\cdot$ QR $=$ QR; QR $\cdot$ QNR $=$ QNR; QNR $\cdot$ QNR $=$ QR.

In order for Alice to cheat and in order to foil the attack, one must be able to distinguish between QR and QNR mod p (at least for prime p) quickly. This has been known for 200 years.

Quadratic residues

Theorem. For prime $p > 2$ and $0 < a < p$, the congruence $x^2 \equiv a \pmod{p}$ has exactly 2 solutions (in a CSR) if a is a QR mod p and no solution if a is a QNR mod p.

Theorem. For prime $p > 2$, there are $(p-1)/2$ QR and $(p-1)/2$ QNR in a CSR (or RSR) mod p

Theorem. For prime $p > 2$ and $0 < a < p$, $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$.

Theorem. (The Euler Criterion.) For prime $p > 2$ and $0 < a < p$,
$a^{(p-1)/2} \equiv 1$ if a is a QR mod p and
$a^{(p-1)/2} \equiv -1$ if a is a QNR mod p.