

### Exponentiation ciphers

Choose a large integer  $n$  for modulus. Encode plaintext as (blocks)  $0 \leq M < n$ . 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  $0 \leq M < n$ , 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  $\phi(n)$ .

### Pohlig-Hellman cipher

This is NOT a public-key cipher.

Let  $n = p = \text{prime}$ . 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  $M \equiv C^d \pmod{p}$ .

The Discrete Logarithm Problem, given positive integers  $a$ ,  $b$  and  $m$ , find  $x$  so that  $a^x \equiv b \pmod{m}$ , is a well known hard problem in Number Theory. Although there are some easy cases, such as  $m = p = \text{prime}$  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  $m$ .

## 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  $ed \equiv 1 \pmod{(p-1)(q-1)}$ .

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 200 decimal digits, this is too hard for current algorithms and computers.

## The El-Gamal public-key cipher

The **ElGamal public key cryptosystem** is defined as follows: Fix a large prime  $p$  which is public. Also public is a primitive root  $g$  modulo  $p$  in  $1 < g < p$ . 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

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

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  $M$ :

$$(g^{a_A k})^{-1} (Mb_A^k) \equiv (g^{a_A k})^{-1} (Mg^{a_A k}) \equiv M \pmod{p}.$$

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

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

#### 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 decipheres 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.

## 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 Massey-Omura:  $E_A$ ,  $D_B$ , 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  $M$ .