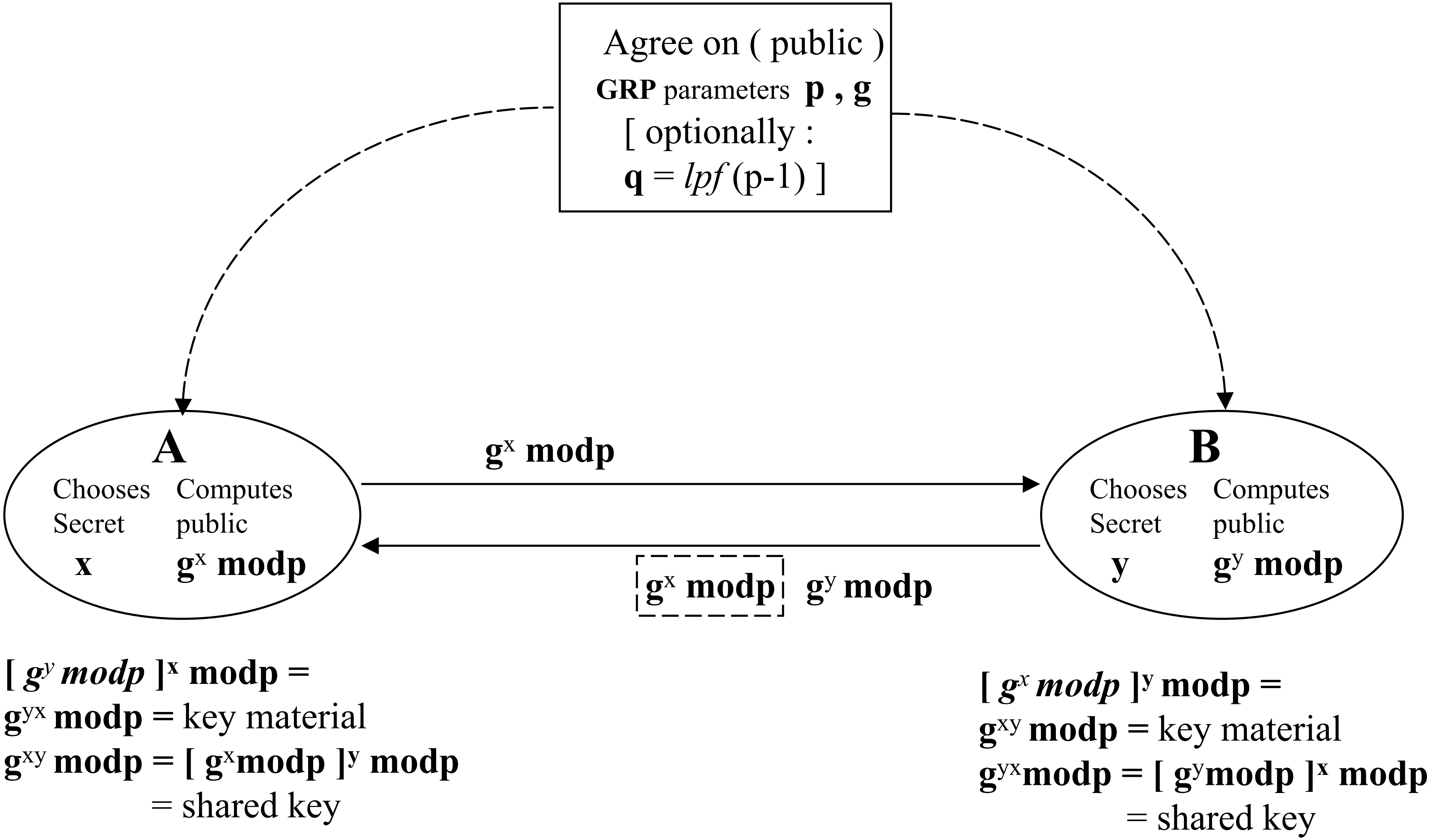
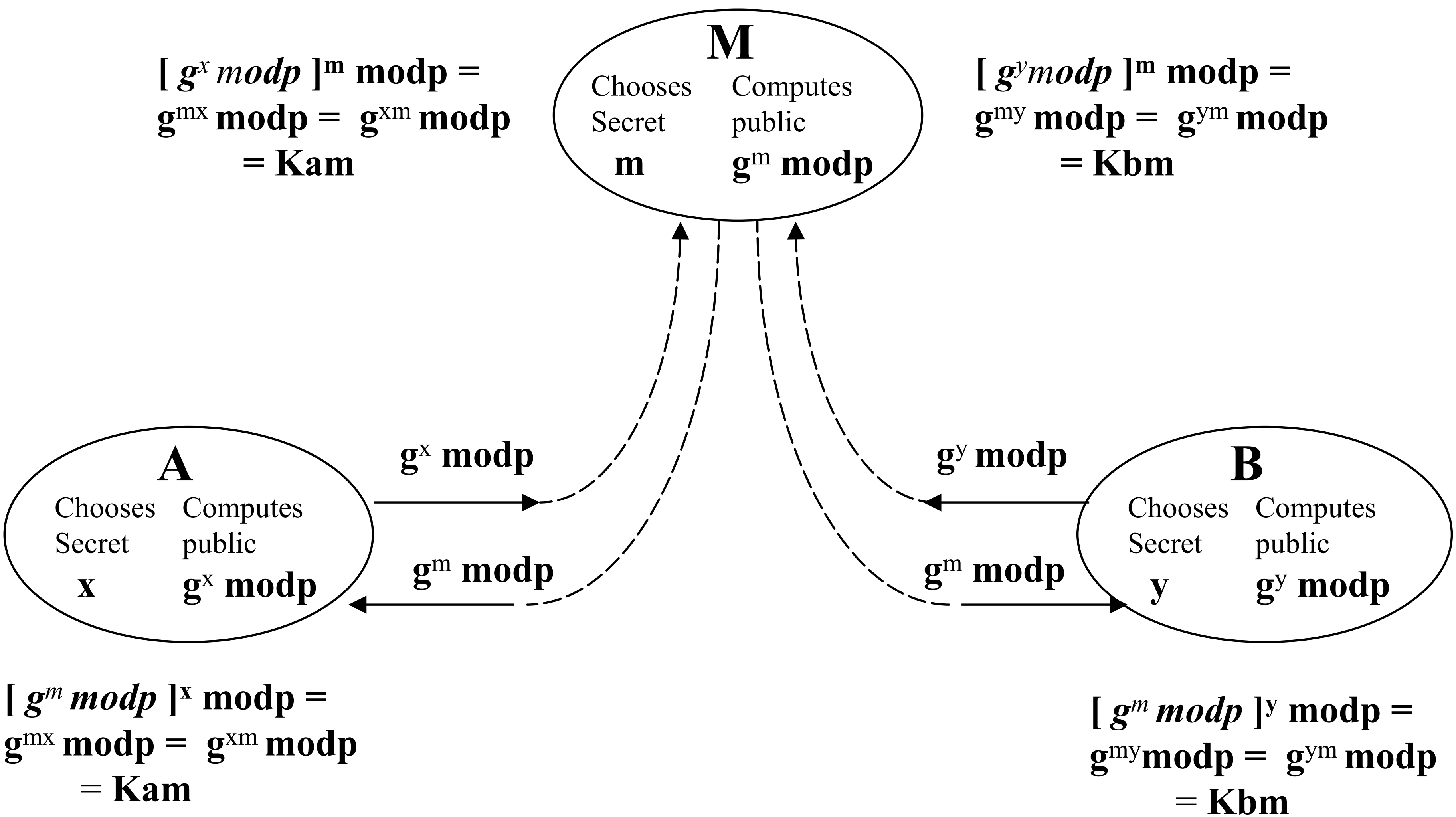
Diffie-Hellman Key-Exchange Protocol



Shared key determination is based on the computational complexity of finding **x (y),** given **g, p,** g^x **modp (** g^y **modp)**; i.e., of computing discrete logarithms.

Man-inthe-Middle Attack => no Authentication



Problem 1 : Key Exchange without Authentication

Probelm 2: Reuse of x, y => replay and forced reuse of shared key; timing attack

Potential Solutions
(not mutually exclusive)

1. Secure, published associations : A < - > (g_A, p_A, g_A^x modp_A)

   = > equivalent of using signed, public-key certificates

2. Establish secure dependency of *key exchange* on *prior, independent authentication*

   = > use of other keys for mutual authentication

3. Establish private, shared groups (g , p: q) between two communicating parties

   = > use of independent protocols for group sharing, privacy
   (separate multicast groups)

4. Use explicit replay-detection mechanisms; e.g., nonces (and PK encryption)

Note: Potential solutions depend on other security protocols

Discrete Logarithms (aka. indices)

1. Primitive roots of modulus p

- let **g** and **p** be *relatively prime* (note: **p** does *not* have to be a prime number)

- consider all **m** for which $g^m \equiv 1 \bmod p$

  o minimum **m** is the order of **g modp**,
  the length of period generated by **g**
  the exponent to which **g** belongs (**modp**)

  o maximum $m = \phi(p)$, by Euler's theorem, where $\phi(p)$ is the *totient* of **p**

- if **g** is of the order $\phi(p)$, then **g** is *a primitive root* of **p,** which means that:

  $g^1 \bmod p$, $g^2 \bmod p$, ... , $g^{\phi(p)} \bmod p$
  - are distinct and represent a permutation of { **1**, ..., **p-1** }
  - are relatively prime to **p**
  - if **p** is prime, $\phi(p) = p-1$; so the set size (length of period) is **p-1**

Note: the only integres with primitive roots are those of the form
2, 4, p^a, $2p^a$ where **p** is any (odd) prime

Discrete Logarithms (aka. indices) -ctnd

2. Properties of Discrete Logarithms

Observation

o any integer $\mathbf{x = r \bmod p}$ for any **r , p** where $\mathbf{0 \le r \le p-1}$

o if **g** is a primitive root of *prime* **p** , $\mathbf{x = g^i \bmod p}$**,** where $\mathbf{0 \le i \le p-1}$

Definition

o exponent **i** is the *index (discrete log)* of **x** in *base* **g modp**; i.e., $\mathbf{ind_{g,p}(x)}$

Ordinary Logarithms

1. **Definition :** $\mathbf{x = b^{\log_b(x)}}$
2. $\mathbf{\log_b(1) = 0}$
3. $\mathbf{\log_b(b) = 1}$
4. $\mathbf{\log_b(ab) = \log_b(a) + \log_b(b)}$

4a. $\mathbf{\log_b(a^r) = r \times \log_b(a)}$

Discrete Logarithms

1. **Definition :** $\mathbf{x = g^{ind_{g,p}(x)}}$
2. $\mathbf{ind_{g,p}(1) = 0}$
3. $\mathbf{ind_{g,p}(g) = 1}$

4.* $\mathbf{ind_{g,p}(xy) = [\, ind_{g,p}(x) + ind_{g,p}(y)\,] \bmod \phi(p)}$

4a. $\mathbf{ind_{g,p}(x^r) = r \times [\, ind_{g,p}(x)\,] \bmod \phi(p)}$

* Proof: $g^{ind_{g,p}(xy)} \bmod p = (g^{ind_{g,p}(x)} \bmod p)\,(g^{ind_{g,p}(y)} \bmod p)\underbrace{(g^{k\,\phi(p)} \bmod p)}_{=1}$

$$= [\, g^{ind_{g,p}(x) + ind_{g,p}(y) + k\,\phi(p)}\,] \bmod p$$

Hence, $ind_{g,p}(xy) = [\, ind_{g,p}(x) + ind_{g,p}(y)\,] \bmod \phi(p)$ since any $z = q + k\,\phi(p)$ can be written as $z = q \bmod \phi(p)$

Cryptographic Strength

1. Stong Primes (i.e., Sophie-Germaine) primes

- o $P = 2Q + 1$, where P, Q = primes; Q = Largest Prime Factor (lpf) of P

2. Schnorr subgroups

- o $P = kQ+1$, where k may be small
- o Generation and Validation of Group Choices
  - Estimate on 25 MHZ RISC or 66 MHZ CISC
  - Generation of P, k, Q => about 10 minutes for a group of 2^{1024} elements
  - Validation => 1 minute

3. Key Length Estimates

- o practical level of security: 75 bits => Q = lpf(P) = 150 bits => P = > 980 bits
- o size of exponent should be at least 2 x length of key = 2 x 75 = 180 bits

- o 20 year security: 90 bits => Q = lpf(P) = 180 bits => P = > 1400 bits
- o size of exponent should be at least 2 x length of key = 2 x 90 = 180 bits

- o extended security: 128 bits => Q = lpf(P) = 256 bits => P = > 3000 bits
- o size of exponent should be at least 2 x length of key = 2 x 128 = 156 bits

4. **Reuse of x (e.g., more than 100 times)** => timing attacks on x; use "blinding factor" r

- o $A = (r g^y)$, where r is a random group element
- o $B = A^x = (r g^y)^x = (r^x)(g^{xy})$
- o $C = B (r^{-x}) = (r^x)(r^{-x}) (g^{xy}) = g^{xy}$

Group Descriptors - 2 Examples

Group Type: *MODP* /* modular exponentiation group, mod P*/
Size of Field (in bits): $\lceil \log_2 P \rceil$ a 32-bit integer
Defining Prime P: a multi-precision integer
Generator G: a multi-precision integer $2 \leq G \leq P-2$
optional:
Largest prime factor of P-1 : the multiprecision integer Q
Strength of Group: a 32-bit integer (approx. the no. of key bits protected; $\log_2$ of workfactor)

Group Type: *ECP* /* elliptic curve group, mod P */
Size of Field (in bits): $\lceil \log_2 P \rceil$ a 32-bit integer
Defining Prime P: a multi-precision integer
Generator (X, Y): two multi-precision integers $(X, Y \leq P)$
Parameters of the curve A, B: two multi-precision integers $(A, B \leq P)$
optional:
Largest prime factor of group order : the multi-precision integer
Order of the group: a multi-precision integer
Strength of Group: a 32-bit integer (approx. the no. of key bits protected; $\log_2$ of workfactor)

elliptic curve equation: $Y^2 = X^3 + AX + B$