INFORMATION AND NETWORK SECURITY (2170709)

PUBLIC KEY CRYPTOGRAPHY

Reference Books

1. **Cryptography And Network Security, Principles And Practice Sixth Edition, William Stallings, Pearson**
2. Information Security Principles and Practice By Mark Stamp, Willy India Edition
3. Cryptography & Network Security, Forouzan, Mukhopadhyay, McGrawHill
4. Cryptography and Network Security Atul Kahate, TMH
5. Cryptography and Security, C K Shyamala, N Harini, T R Padmanabhan, Wiley-India
6. Information Systems Security, Godbole, Wiley-India
7. Information Security Principles and Practice, Deven Shah, Wiley-India
8. Security in Computing by Pfleeger and Pfleeger, PHI
9. Build Your Own Security Lab : A Field Guide for network testing, Michael Gregg, Wiley India

Information Security

Information Security is the process of protecting the availability, privacy, and integrity of data.

(wisegeek.org)

Information Security is the practice of defending information from unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction.

(wikipedia.org)

Basic Terms

- Plaintext
  - An original message
- Ciphertext
  - The coded message
- Encryption
  - The process of converting from plaintext to ciphertext
- Decryption
  - The process of converting from ciphertext to plaintext

Basic Terms

- Cryptography
  - The techniques used for encrypting the message fall into the area of cryptography.
- Cryptanalysis
  - The techniques used for decrypting the message fall into the area of cryptanalysis.
- Cryptology
  - The areas of cryptography and cryptanalysis together are called cryptology.

Symmetric/Conventional Encryption

- Uses a ***single*** key
- Shared by both sender and receiver

Symmetric/Conventional Encryption

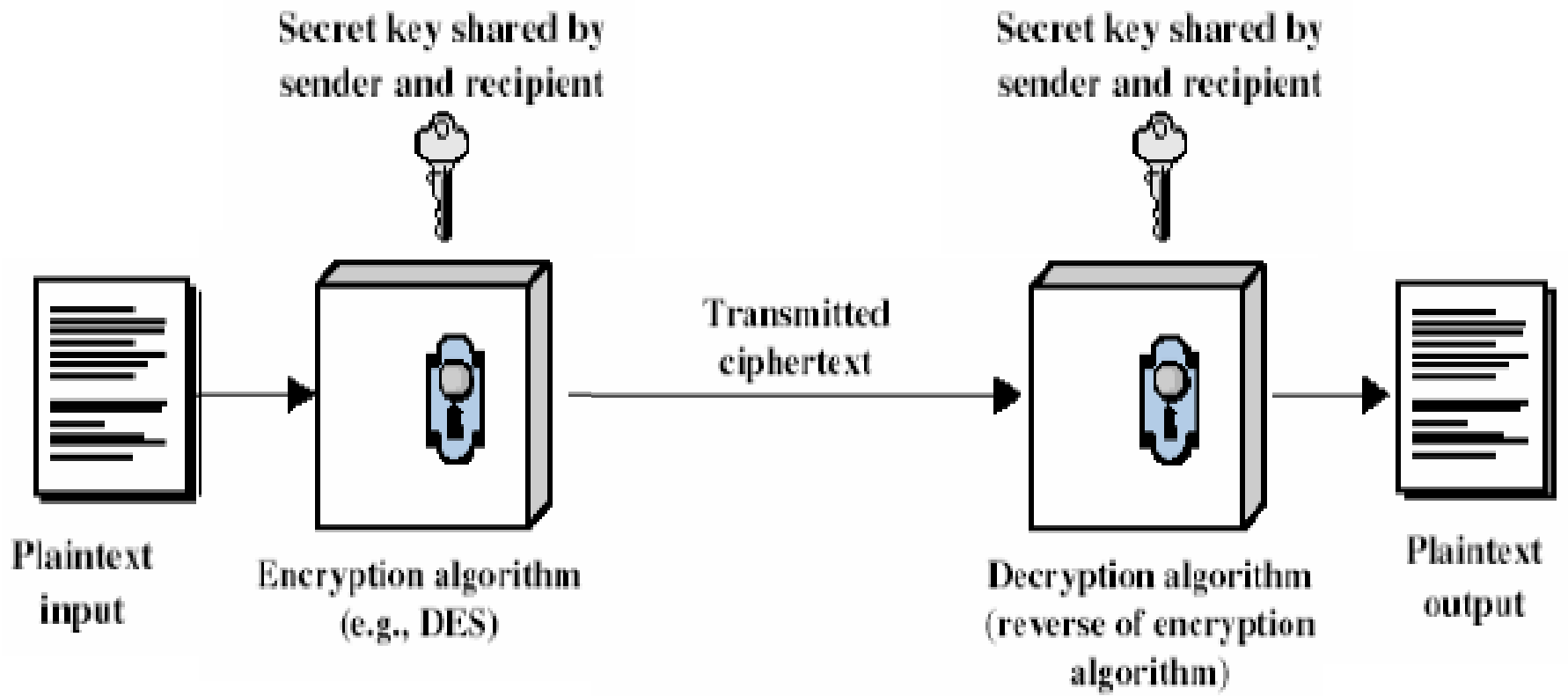


Fig. Simplified Model of Symmetric Encryption

Symmetric/Conventional Encryption

- Two requirements
  - 1. Need of a strong algorithm
  - 2. Secret key must be shared in a secure manner
- If the key is disclosed, communications are compromised.
- It is ***symmetric***, parties are equal. Hence does not protect sender from receiver forging a message & claiming is sent by sender.

Public Key Cryptography

- Principles of Public Key Cryptography:
- Uses two different keys – ***Public key & Private key***
- Not possible to get decryption key, knowing only the algorithm and encryption key
- 6 ingredients:
  - Plaintext
  - Encryption algorithm
  - Public Key
  - Private key
  - Ciphertext
  - Decryption algorithm

Public Key/Asymmetric Encryption

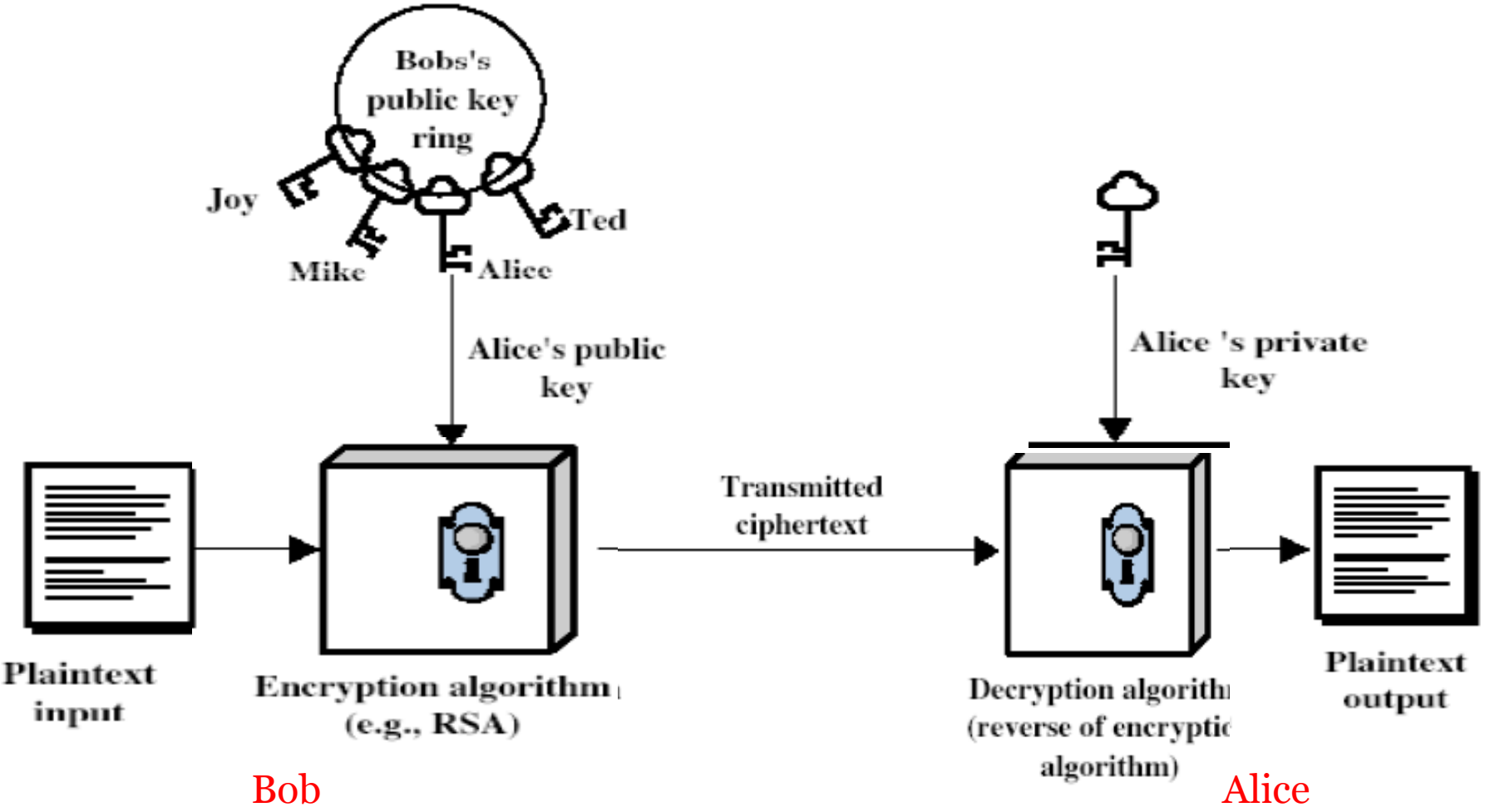


Fig. Asymmetric/Public Key Encryption

Public Key Cryptography

- Each user generates a pair of keys
- One key is placed in a public register and other is kept secret
- Public Key
  - may be known by anybody, and can be used to encrypt messages, and verify signatures.
- Private Key
  - known only to the recipient, used to decrypt messages, and sign (create) signatures.

Public Key Cryptography

- Message is ***encrypted*** using receiver's ***public key***
- Receiver ***decrypts*** the message using its ***private key***
- Asymmetric because
  - those who encrypt messages cannot decrypt messages
  - those who verify signatures cannot create signatures

Public Key Cryptography

- How to verify a message comes infect from the claimed sender??
- Requirement of Public Key Authentication – ***Digital Signature***
- Sender *encrypts* a message with its *private key*
- Receiver *decrypts* the message using sender's *public key*

Public Key Authentication

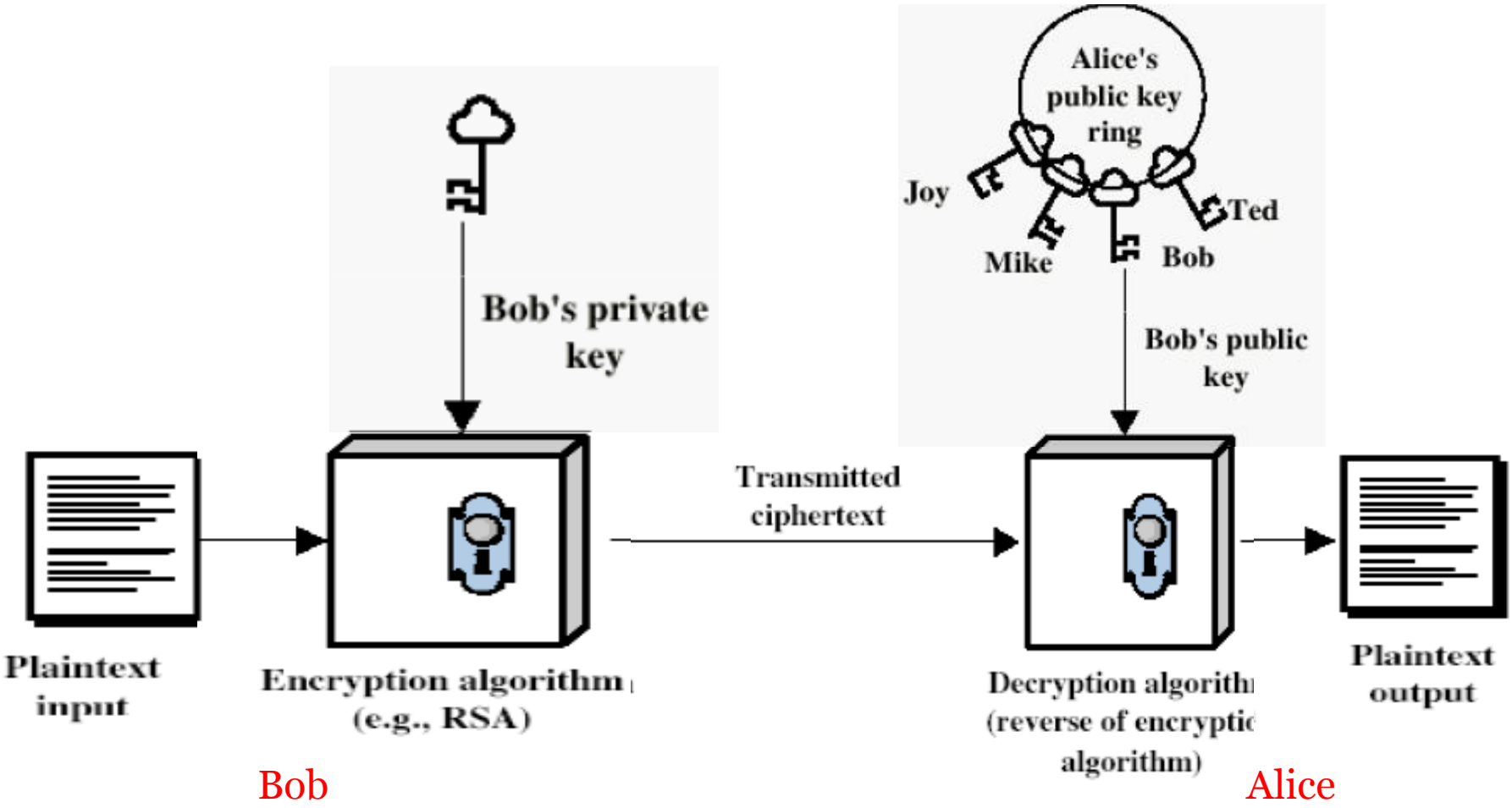


Fig. Asymmetric/Public Key Authentication

Public Key Cryptography

- Only authentication does not provide confidentiality of message
- Any other receiver can decrypt the message using sender's public key

Public Key Authentication and Confidentiality

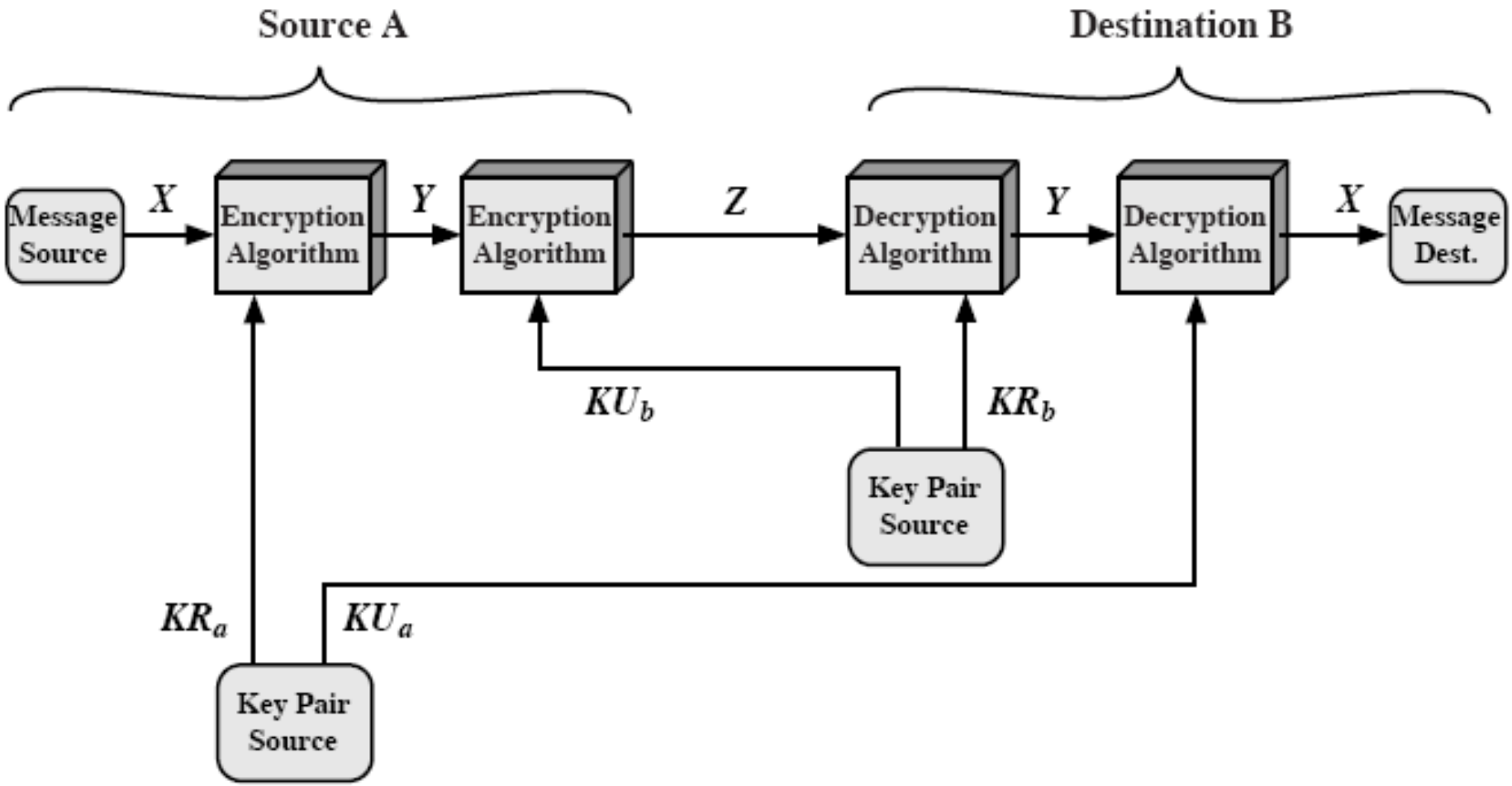


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

Public Key Cryptography

- Confidentiality
  - $C = ENC_{KUb}[P] = E_{PUb}[P]$
  - $P = DEC_{KRb}[C] = D_{PRb}[C]$
- Authentication/Digital Signature
  - $C = ENC_{KRa}[P] = E_{PRa}[P]$
  - $P = DEC_{KUa}[C] = E_{PUa}[C]$
- Confidentiality + Authentication
  - $C = ENC_{KUb}[ENC_{KRa}(P)] = E_{PUb}[E_{PRa}(P)]$
  - $P = DEC_{KUa}[DEC_{KRb}(C)] = E_{PUa}[E_{PRb}(P)]$

Applications of Public Key Cryptography

- Encryption/Decryption
  - Sender encrypts a message with receiver's public key
- Digital Signature
  - Sender signs the message with its private key
- Key Exchange
  - Two parties co-operate to exchange a session key

- RSA algorithm uses all 3 applications
- Diffie-Hellman algorithm uses only key exchange application
- Digital signature standard (DSS) uses only digital signature algorithm

Requirements for Public Key Cryptography

- Computationally easy to generate private-public key pair
- Computationally easy to generate ciphertext

  $C = E_{KUb}[M]$
- Computationally easy to decrypt ciphertext

  $M = D_{KRb}[C]$
- Computationally infeasible to determine private key by just knowing public key
- Computationally infeasible to determine original message by knowing public key and ciphertext

Security of Public Key Schemes

- Like private key schemes, brute force exhaustive search attack is always theoretically possible
- But keys used are too large (>512 bits)

- Requires the use of very large numbers
- Hence is slow compared to private key schemes

RSA Algorithm

- Best known & widely used public-key scheme by Rivest, Shamir & Adleman in 1977
- Can be used to provide both secrecy & digital signatures
- Security due to the cost of factoring large numbers

RSA Algorithm

- RSA Contents:
  - Key Generation
  - Encryption
  - Decryption
  - Digital Signature
  - Signature Verification

RSA Algorithm

- 1. Key Generation:
- Each user generates a public/private key pair by:
  - Selecting two large primes at random: p, q
  - Computing their system modulus $N=p.q$
  - Note $\phi(N)=(p-1)(q-1)$

- Selecting at random the encryption key e
  - Where $1<e<\phi(N)$, $\gcd(e,\phi(N))=1$

RSA Algorithm

- Solve following equation to find decryption key *d*
  - $e.d=1 \bmod \phi(N)$ and $0<=d<=N$

- Publish their public encryption key: $PU=\{e, N\}$
- Keep secret private decryption key: $PR=\{d, p, q\}$

- It is critically important that the factors p & q of the modulus N are kept secret

RSA Algorithm

- 2. Encryption:
- Let sender *A* wants to send a secret message *M* to receiver *B*

- To encrypt a message *M*, the sender *A* obtains public key of recipient *B*: $PU_b=\{e, N\}$

- Sender *A* represents the message *M* as a positive integer.

- Seder *A* computes:
  - $C=M^e \bmod N$, where $M<N$ (send *C* to receiver *B*)

RSA Algorithm

- 3. Decryption:
- To decrypt the ciphertext C, receiver B uses its private key $PR_b=\{d, p, q\}$

- Receiver B computes:
  - $M=C^d \bmod N$

- Note that the message M must be smaller than the modulus N.

RSA Algorithm

- 4. Digital Signature:
- Sender *A* does the following:
- Create a message digest (summary) of the information to be sent

- Represent the message digest as an integer *m* between 0 to *N-1*

- Uses its private key $PR_a=\{d, p, q\}$ to compute the signature
  - $S = m^d \bmod N$ (Send *S* to recipient *B*)

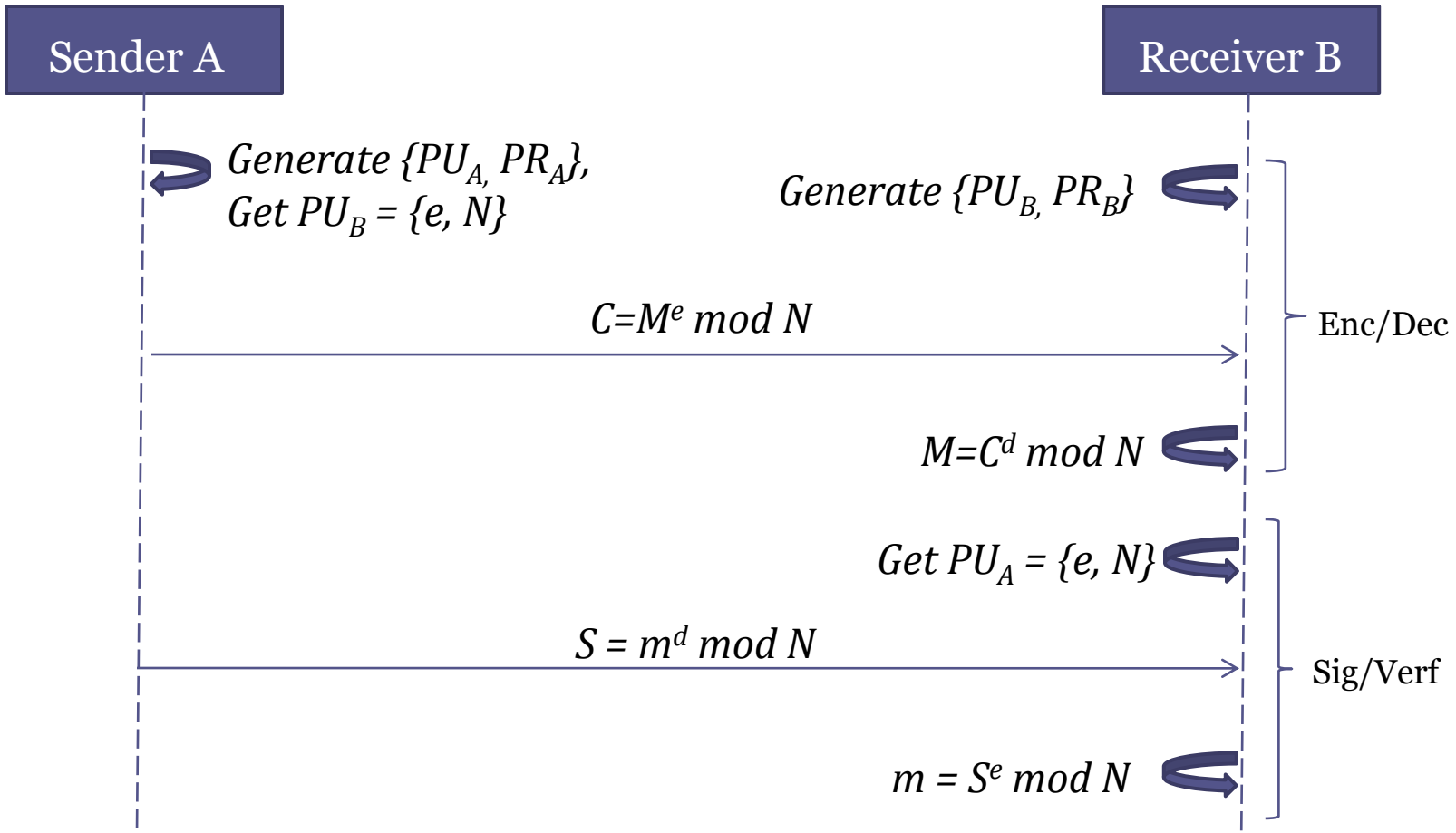
RSA Algorithm

- 5. Signature Verification:
- Receiver *B* does the following:

- Use sender *A*'s public key $PU_a=\{e, N\}$ to compute the integer
  - $m = S^e \bmod N$

- Extract the message digest from this integer *m*.

RSA Algorithm



Why RSA Works?

- To encrypt a message
  - $C = M^e \bmod N$
- To decrypt a message
  - $M = C^d \bmod N$
  - $= (M^e \bmod N)^d \bmod N$
  - $= M^{ed} \bmod N$
  - $= M^{1+k\phi(N)} \bmod N$
  - $= M$

- By corollary to Euler's theorem, given two prime numbers p and q & two integers N and M such that $N = pq$ and $0<M<N$ and any arbitrary integer k, we have
  - $M^{k\phi(N)+1} Mod\ N = M^{k(p-1)(q-1)+1} \equiv M \bmod N$

Recommended Key Size of RSA

- With a key of length *n* bits, there are 2^n possible keys.
- The effectiveness of public key cryptosystems depends on the intractability of certain mathematical problems such as integer factorization.
- RSA claims that 1024-bit keys are likely to become crackable some time between 2006 and 2010 and that 2048-bit keys are sufficient until 2030.
- An RSA key length of 3072 bits should be used if security is required beyond 2030.

RSA Example-1

- Select primes
  - *p=11 & q=3*
- Compute *n*
  - *N = pq = 11 x 3 = 33*
- Compute ø(*n)* value
  - *ø(N)=(p–1)(q-1)=10 x 2 = 20*
- Select encryption parameter
  - *e*: gcd(*e*,20)=1; choose *e = 3*
- Determine decryption parameter
  - *d*: *de=1 mod 20 and d < 20 (d=[1+(k*ø(N))]/e)*
  - Value is *d*=7 since 7*3=21=20*1+1

- Publish public key *PU*={3, 33}
- Keep secret private key *PR*={7, 11, 3}

RSA Example-1

- Given message $M = 7$ (note that 7<33)
- Encryption is
  - $C = 7^3 \bmod 33$
  - $= 343 \bmod 33$
  - $= 13$
- Decryption is
  - $M = 13^7 \bmod 33$
  - $= 13^{(3+3+1)} \bmod 33$
  - $= (13^3 \bmod 33)(13^3 \bmod 33)(13 \bmod 33) \bmod 33$
  - $= (2197 \bmod 33)(2197 \bmod 33)(13) \bmod 33$
  - $= 19*19*13 \bmod 33 = 4693 \bmod 33$
  - $M = 7$

RSA Example-2

- Select primes
  - *p=5 & q=7*
- Compute *n*
  - *N = pq = 5 x 7 = 35*
- Compute ø(*n)* value
  - *ø(N)=(p–1)(q-1)=4 x 6 = 24*
- Select encryption parameter
  - *e*: gcd(*e*,24)=1; choose *e = 5*
- Determine decryption parameter
  - *d*: *de=1 mod 24 and d < 35 (d=[1+(k*ø(N))]/e)*
  - Value is *d*=29 since 1+6*24=145/5=29

- Publish public key *PU*={5, 35}
- Keep secret private key *PR*={29, 5, 7}

RSA Example-2

- Given message $M = 5$ (note that 5<35)

- Encryption is
  - $C = 5^5 \bmod 35$
    $= 3125 \bmod 35$
    $= 10$

- Decryption is
  - $M = 10^{29} \bmod 35$
    $= [(10^5 \bmod 35)\ (10^5 \bmod 35)\ (10^5 \bmod 35)$
    $(10^5 \bmod 35)\ (10^5 \bmod 35)\ (10^4 \bmod 35)] \bmod 35$
    $= [(5)(5)(5)(5)(5)(25)] \bmod 35$
    $= 5$

RSA Example-3

- Select primes
  - *p=17 & q=11*
- Compute n
  - *N = pq =17 x 11=187*
- Compute ø(*n)* value
  - *ø(N)=(p–1)(q-1)=16 x 10=160*
- Select encryption parameter
  - *e: gcd(e,160)*=1; choose *e=7*
- Determine decryption parameter
  - d: *ed=1 mod 160 and d < 160*
- Value is d=23 since 23x7=161= 10x16+1

- Publish public key PU={7,187}
- Keep secret private key PR={23,17, 11}

RSA Example-3

- Given message $M = 88$ (note that 88<187)

- Encryption is
  - $C = 88^7 \bmod 187$
  - $= 88^{(3+3+1)} \bmod 187$
  - $= (88^3 \bmod 187)(88^3 \bmod 187)(88 \bmod 187) \bmod 187$
  - $= (44 * 44 * 88) \bmod 187$
  - $= 11$

- Decryption is
  - $M = 11^{23} \bmod 187 = 88$

RSA Ciphertext Values

- Ciphertext for all the possible values of m from (0 to 32):

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
c	0	1	8	27	31	26	18	13	17	3	10	11	12	19	5

m	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
c	9	4	29	24	28	14	21	22	23	30	16	20	15	7	2

m	30	31	32
c	6	25	32

- Some of the ciphertexts are unconcealed messages because *N* selected is small.
- With larger values of *N*, it isn't a problem.

RSA Example-4

- Consider the text grouping in the groups of three i.e.
  - ATTACKXATXSEVEN = ATT ACK XAT XSE VEN

- Represent the blocks in base 26 using A=0, B=1, C=2…..

- ATT = $0 * 26^2 + 19 * 26^1 + 19 * 26^0 = 513$
- ACK = $0 * 26^2 + 2 * 26^1 + 10 * 26^0 = 62$
- XAT = $23 * 26^2 + 0 * 26^1 + 19 * 26^0 = 15567$
- XSE = $23 * 26^2 + 18 * 26^1 + 4 * 26^0 = 16020$
- VEN= $21 * 26^2 + 4 * 26^1 + 13 * 26^0 = 14313$

- What should be the value of *N* ?

- The value of *N* should be greater than 17575. because maximum *M* value can be ZZZ = 17575 and *N* should be greater than *M*.

RSA Example-4

- Let p = 137 and q = 131; so that $N = pq$ = 17947
- Compute phi ø(N) value
  - $ø(N)=(p–1)(q-1)=136 \times 130 = 17680$
- Select encryption parameter e:
  - gcd(e,17680)=1; choose $e = 3$
- Determine decryption parameter
  - d: $de=1 \bmod 17680$ and $d < 17680$
  - Value is d=11787 since $11787 \times 3=35361=17680 \times 2+1$
- Publish public key PU={3, 17947}
- Keep secret private key PR={11787, 137, 131}

RSA Example-4

- Given message
  - M = ATT = 513
- Encryption is
  - $C = 513^3 \bmod 17947 = 8363$
- Decryption is
  - $M = 8363^{11787} \bmod 17947 = 513$

Computational aspects of RSA

- Exponentiation in Modular Arithmetic
  - $(a*b) \bmod n = [(a \bmod n)*(b \bmod n)] \bmod n$
  - $x^{11} \bmod n = [(x^1 \bmod n)*(x^2 \bmod n)*(x^8 \bmod n)] \bmod n$ or
  - $x^{16} \bmod n = x^2, x^4, x^8, x^{16} \bmod n$ (repeatedly take square of each partial result)

Modular Exponentiation: Square & Multiply

- A fast, efficient algorithm for exponentiation
- Repeated squaring
- Ex: $2003^{17} \bmod 3713$

$$
\begin{aligned}
2003^{17} &\pmod{3713} \\
&\equiv 2003^{1} \times 2003^{16} \pmod{3713} \\
&\equiv 2003 \times 3157 \pmod{3713} \\
&\equiv 6{,}323{,}471 \pmod{3713} \\
&\equiv 232 \pmod{3713}
\end{aligned}
$$

Term	Compute	mod 3713
2003^{1}	2003	2003
2003^{2}	2003^{2}	1969
2003^{4}	1969^{2}	589
2003^{8}	589^{2}	1612
2003^{16}	1612^{2}	3157

Computational aspects of RSA

- Efficient operation using the Public Key
  - Common choices of public key *e* are 3, 17, 65537
  - Each choice has only two 1 bits, so the number of multiplications required to perform exponentiations are minimized.
- Efficient operation using the Private Key
  - A small value of private key *d* is vulnerable to a brute force attack
- Key Generation
  - Primes *p* and *q* must be large numbers

Security of RSA

- Possible attacking approaches to RSA
  - Brute Force attack
    - Trying all possible private keys
  - Mathematical attacks
    - Several approaches for factoring the product of two primes
  - Timing attacks
    - Depend on the running time of decryption algorithm
  - Chosen Ciphertext attacks
    - Exploits the properties of RSA algorithm

Security of RSA

- Brute Force Attack
  - Use a large key space to defense against brute force attack.
  - The larger the number of bits in *d*, the better.
  - The larger the size of the key, the slower the system will work.

Security of RSA

- Mathematical Attack: Factoring problem
  - To attack RSA mathematically,
    - Factor n into its two prime factors. This enables calculation of $\phi(n)=(p-1)(q-1)$, which enables determination of e and then $d \equiv e^{-1} \bmod \phi(n)$
    - Determine $\phi(n)$ directly without first determining p and q. Again this enables determination of e and then $d \equiv e^{-1} \bmod \phi(n)$
    - Determine d directly without first determining $\phi(n)$.

Security of RSA

- Mathematical Attack: Factoring problem
  - For a large n with large prime factors, factoring is a hard problem.
  - Algorithm's inventors suggested following constraints on p and q:
    - p and q should differ in length by only a few digits
    - Both $(p-1)$ and $(q-1)$ should contain a large prime factor
    - GCD$(p-1, q-1)$ should be small
  - If $e<n$ and $d<n^{1/4}$, then d can be easily determined.

Security of RSA

- Timing Attacks
  - Determination of private key by keeping track of how long it takes to decrypt messages ($M = C^d \ mod \ N$)

- Constraints to prevent timing attacks
  - Constant exponentiation time
    - Ensure that all exponentiations take the same amount of time before returning a result
    - Simple but does degrade the performance
  - Random Delay
    - Add a random delay to the exponentiation algorithm to confuse the timing attack

Security of RSA

- Constraints to prevent timing attacks
  - Blinding
    - Multiply the ciphertext by a random number before performing exponentiation.
    - Prevents the attacker from knowing what ciphertext bits are processed and therefore prevents bit-by-bit analysis

Security of RSA

- Constraints to prevent timing attacks
  - Blinding
    - The private key operation $M = C^d \bmod N$ is performed as follows:

      1. Generate a secret random number r between 0 and N-1
      2. Compute $C' = C\,(r^e) \bmod N$
      3. Compute $M' = (C')^d \bmod N$
      4. Compute $M = (M')\ r^{-1} \bmod N$ (r^{-1} is multiplicative inverse of $r \bmod N$ and $r^{ed} \bmod N = r \bmod N$)

Security of RSA

- Chosen Ciphertext Attack (CCA)
  - CCA is defined as an attack in which adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key.
  - Thus, the adversary tries to gain the knowledge of private key that is used to decrypt the ciphertexts.

Security of RSA

- Chosen Ciphertext Attack (CCA)
  - Property of RSA:

$$E(PU, M_1) * E(PU, M_2) = E(PU, [M_1 * M_2])$$

  - We can decrypt $C = M^e \bmod N$ using CCA as follows:
    - Compute $X = (C * 2^e) \bmod N$
    - Submit X as a chosen ciphertext and receive back $Y = X^d \bmod N$
    - But now note that $X = (C \bmod N) * (2^e \bmod N)$

$$= (M^e \bmod N) * (2^e \bmod N)$$

$$= (2M)^e \bmod N$$

    - Therefore $Y = (2M) \bmod N$. from this we can deduce M.

Security of RSA

- Chosen Ciphertext Attack (CCA)
  - To overcome this attack, practical RSA-based cryptosystems randomly pad the plaintext prior to encryption.
  - To modify plaintext, *Optimal Asymmetric Encryption Padding (OAEP)* is used.

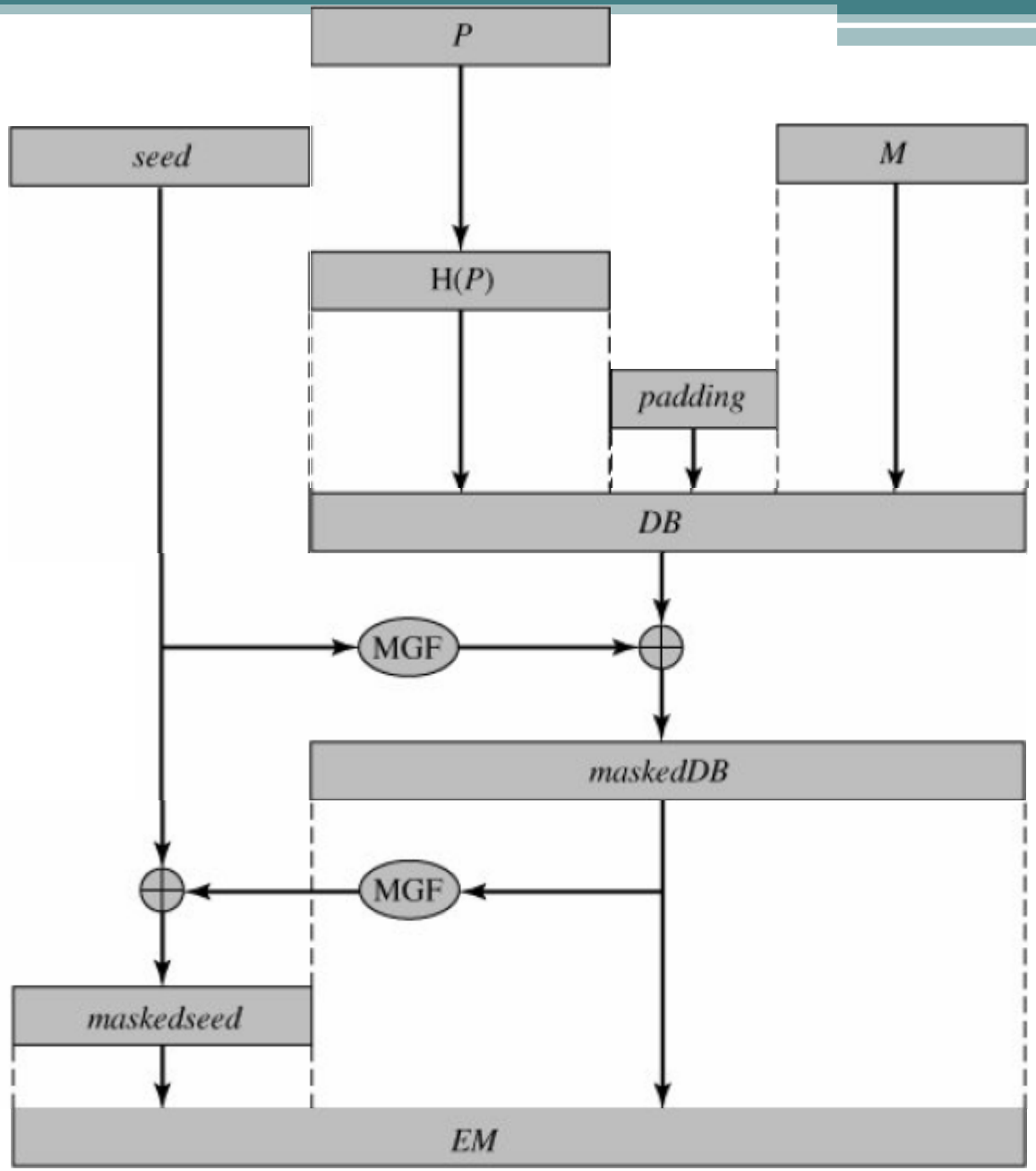


Fig. Encryption using OAEP

Security of RSA

- Working of OAEP
  - Message M is first padded.
  - Optional parameter P is passed through a hash function H.
  - The output of hash, padding bits and M are concatenated to form the overall Data Block (DB).
  - Next, a random seed is generated and passed through another hash function, called mask generation function (MGF) .

(continue…)

Security of RSA

- Working of OAEP
  - Resulting hash value is bit-by-bit XORed with *DB* to produce a *maskedDB*.
  - The *maskedDB* is passed through *MGF* to form a hash that is XORed with seed to produce a *maskedseed*.
  - The concatenation of a *maskedseed* and *maskedDB* forms an encrypted message *EM*.
  - *EM* is then encrypted using RSA.

Relative Prime Numbers

- Two numbers a, b are relatively prime if they have no common divisors apart from 1 (i.e, gcd(a,b)=1)

- Ex:

  8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor.

Euler's Totient Function: ø(n)

- When doing arithmetic modulo n
- *complete set of residues* is: 0..n-1
- *reduced set of residues* is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}

- Number of elements in reduced set of residues is called the ***Euler Totient Function ø(n)***
  - Eg. for n=10, $ø(n) = 4$

Euler's Totient Function: ø(n)

- For p, where p is prime, $\phi(p) = p-1$
- For p.q, where p,q are prime and $p \neq q$,

  $\phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$
- Ex:
  - $\phi(37) = 36$
  - $\phi(21) = \phi(3*7) = \phi(3)*\phi(7) = (3-1)\times(7-1) = 2\times6 = 12$

Primitive Root

- If a and n are relatively prime numbers, a is a primitive root of n if $a, a^2, a^3, \ldots, a^{\phi(n)}$ produce distinct values (mod n) and all values are relatively prime to n.
- Order of a must be equal to $\phi(n)$

- Ex: n=14, $\phi(14)=\{1,3,5,9,11,13\}=6$

a	a	a^2	a^3	a^4	a^5	a^6
1	1	1	1	1	1	1
3	3	9	13	11	5	1
5	5	11	13	9	3	1
9	9	11	1	9	11	1
11	11	9	1	11	9	1
13	13	1	13	1	13	1

- All values are distinct
- All relatively prime to 14
- Order of 3 &5 = 6=$\phi(14)$

Thus, Primitive roots of 14 are a = 3 and 5

Diffie-Hellman Key Exchange Algorithm

- Enables two users to exchange a key securely
- Limited to the exchange of the keys
- Depends on discrete logarithms for its effectiveness

Diffie-Hellman Key Exchange Algorithm

- To define a discrete logarithm
  - Define a primitive root a of a prime number p
  - That is, if a is a primitive root of p, then the values

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

    are distinct and consist of integers from 1 through p-1.

- For any integer b and a primitive root a of a number p, we can find a unique exponent i such that

  $$b \equiv a^i \ (mod\ p)$$

- The exponent i is referred to as the *discrete logarithm* of number b for base a (mod p). $i = \text{dlog}_{a,p}(b)$

Discrete Logarithms/Indices

- Let n=9. $\phi(9)$ = ?
- $\phi(9) = 6$ and a=2 is a primitive root of 9.

- We compute various powers of a
- 2^1=2, 2^2=4, 2^3=8,
- $2^4 \equiv 7 \pmod 9$, $2^5 \equiv 5 \pmod 9$, $2^6 \equiv 1 \pmod 9$
- $i = \text{dlog}_{a,n}(b) = \text{dlog}_{2,9}(b)$

Logarithm(i) :	1	2	3	4	5	6
Number(b) :	2	4	8	7	5	1

- While exponentiation is relatively easy, finding discrete logarithms is generally a hard problem.

Diffie-Hellman Key Exchange Algorithm

- Use two publicly available numbers
  - A prime number q
  - An integer α, that is a primitive root of q
- Let two users A and B want to exchange a secret key
- User A selects a random private integer $X_A < q$ and computes a public value
  - $Y_A = \alpha^{X_A} \bmod q$
- Similarly, user B selects a random private integer $X_B < q$ and computes a public value
  - $Y_B = \alpha^{X_B} \bmod q$

The Diffie-Hellman Key Exchange Algorithm

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

The Diffie-Hellman Key Exchange Algorithm

User A Key Generation

Select private X_A $\quad X_A < q$

Calculate public Y_A $\quad Y_A = \alpha^{X_A} \bmod q$

The Diffie-Hellman Key Exchange Algorithm

User B Key Generation

Select private X_B $\quad X_B < q$

Calculate public Y_B $\quad Y_B = \alpha^{X_B} \bmod q$

Diffie-Hellman Key Exchange Algorithm

- Now, user *A* computes the key as
  - $K = (Y_B)^{X_A} \bmod q$
- And, user B computes the key as
  - $K = (Y_A)^{X_B} \bmod q$
- Above two calculations produce the same key value...!!!

The Diffie-Hellman Key Exchange Algorithm

Generation of Secret Key by User A

$$K = (Y_B)^{X_A} \bmod q$$

Generation of Secret Key by User B

$$K = (Y_A)^{X_B} \bmod q$$

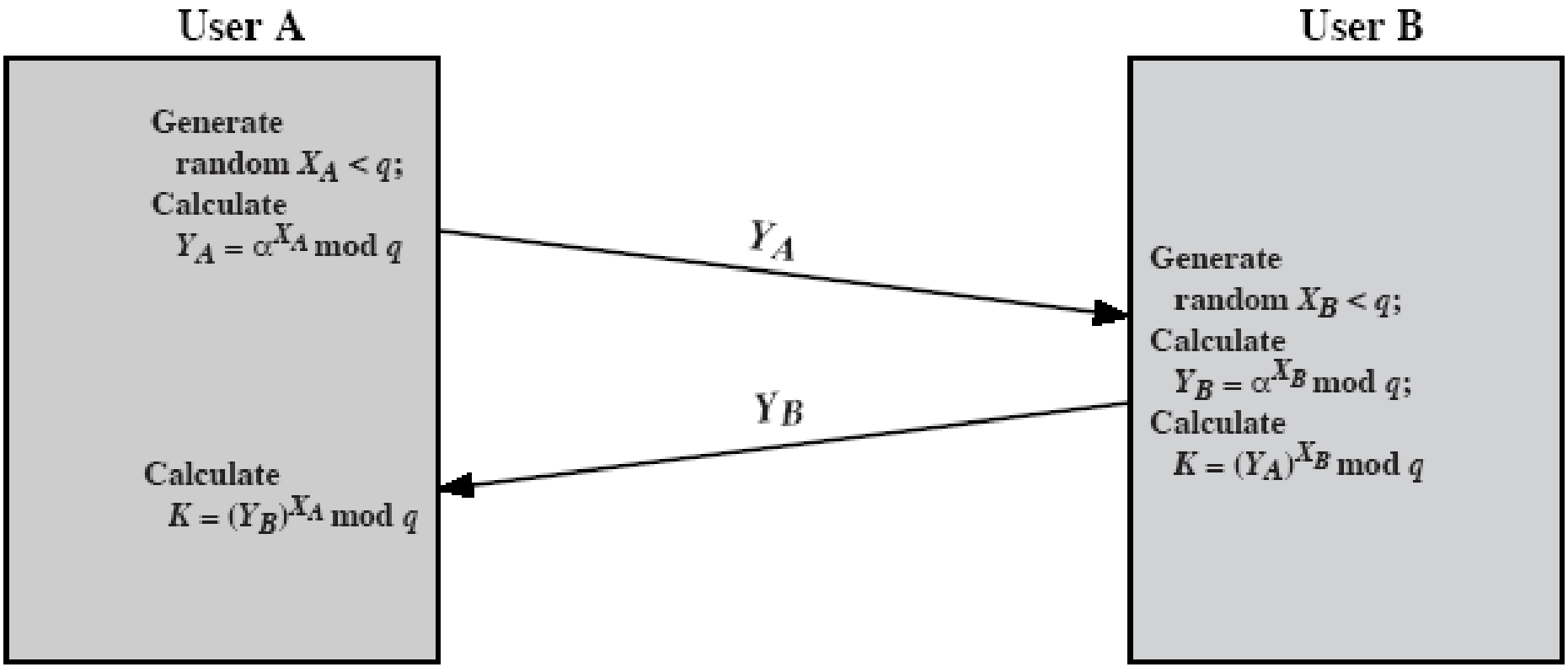
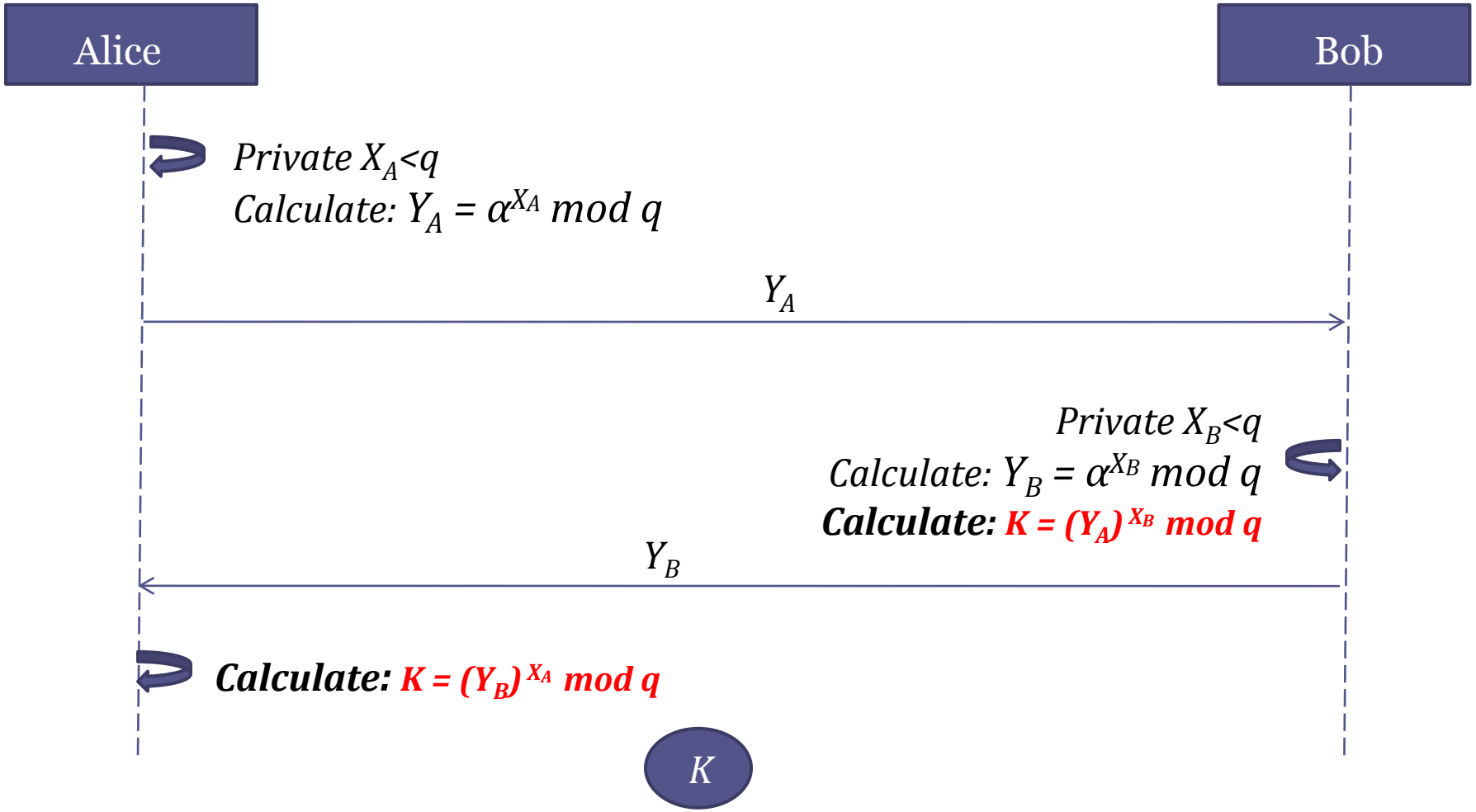


Fig. Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange Algorithm



Diffie-Hellman Key Exchange Algorithm

- Proof:

$$K = (Y_B)^{X_A} \bmod q$$
$$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$$
$$= (\alpha^{X_B})^{X_A} \bmod q$$
$$= (\alpha^{X_A})^{X_B} \bmod q$$
$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$$
$$= (Y_A)^{X_B} \bmod q$$
$$= K$$

- The result is that two sides have exchanged a secret key.

Diffie-Hellman Key Exchange Algorithm

- Because X_A and X_B are private, an opponent only has the knowledge of q, α, Y_A and Y_B
- Thus, for finding secret key X_B of B, the opponent must compute
  - $X_B = \text{dlog}_{\alpha, q}(Y_B)$
- The security of Diffie-Hellman algorithm lies in the fact that it is very difficult to calculate discrete logarithms.

Diffie-Hellman Key Exchange Algorithm

- Example 1:
- Suppose, prime number q = 353
- Primitive root of 353 is α =3

- Suppose A selects a random integer X_A=97 and B selects a random integer X_B =233 as secret keys

- A computes
  - $Y_A = \alpha^{X_A} \bmod q = 3^{97} \bmod 353 = 40$

- B computes
  - $Y_B = \alpha^{X_B} \bmod q = 3^{233} \bmod 353 = 248$

Diffie-Hellman Key Exchange Algorithm

- Now, *A* and *B* compute common secret key *K*.
- A computes
  - $K = (Y_B)^{X_A} \bmod q = 248^{97} \bmod 353 = 160$
- *B* computes
  - $K = (Y_A)^{X_B} \bmod q = 40^{233} \bmod 353 = 160$
- Brute-force attack is possible to determine the secret key *K*, but with large numbers, the attack becomes impractical

Diffie-Hellman Key Exchange Algorithm

- Example 2:
- Suppose, prime number q = 23
- Primitive root of 23 is α =5

- Suppose A selects a random integer X_A=6 and B selects a random integer X_B =15 as secret keys

- A computes
  - $Y_A = \alpha^{X_A} \bmod q = 5^6 \bmod 23 = 8$

- B computes
  - $Y_B = \alpha^{X_B} \bmod q = 5^{15} \bmod 23 = 19$

Diffie-Hellman Key Exchange Algorithm

- Now, *A* and *B* compute common secret key *K*.
- A computes
  - $K = (Y_B)^{X_A} \bmod q = 19^6 \bmod 23 = 2$
- *B* computes
  - $K = (Y_A)^{X_B} \bmod q = 8^{15} \bmod 23 = 2$

Diffie-Hellman Key Exchange Algorithm

- Man-in-the-middle Attack:
- Diffie-Hellman algorithm is insecure against a Man-in-the-middle attack

- Suppose *Alice* and *Bob* wish to exchange a secret key, and *Eve* is an adversary

- The attack proceeds as follows:
  - *Eve* generates two random private keys X_{E1} and X_{E2}, and then computes corresponding public keys Y_{E1} and Y_{E2}
  - *Alice* transmits Y_A to *Bob*

Diffie-Hellman Key Exchange Algorithm

- Man-in-the-middle Attack:
  - *Eve* intercepts Y_A and transmits Y_{E1} to *Bob*. *Eve* also calculates $K2 = (Y_A)^{X_{E2}} \bmod q$
  - *Bob* receives Y_{E1} and calculates $K1 = (Y_{E1})^{X_B} \bmod q$
  - *Bob* transmits Y_B to *Alice*
  - *Eve* intercepts Y_B and transmits Y_{E2} to *Alice*. *Eve* calculates $K1 = (Y_B)^{X_{E1}} \bmod q$
  - *Alice* receives Y_{E2} and calculates $K2 = (Y_{E2})^{X_A} \bmod q$

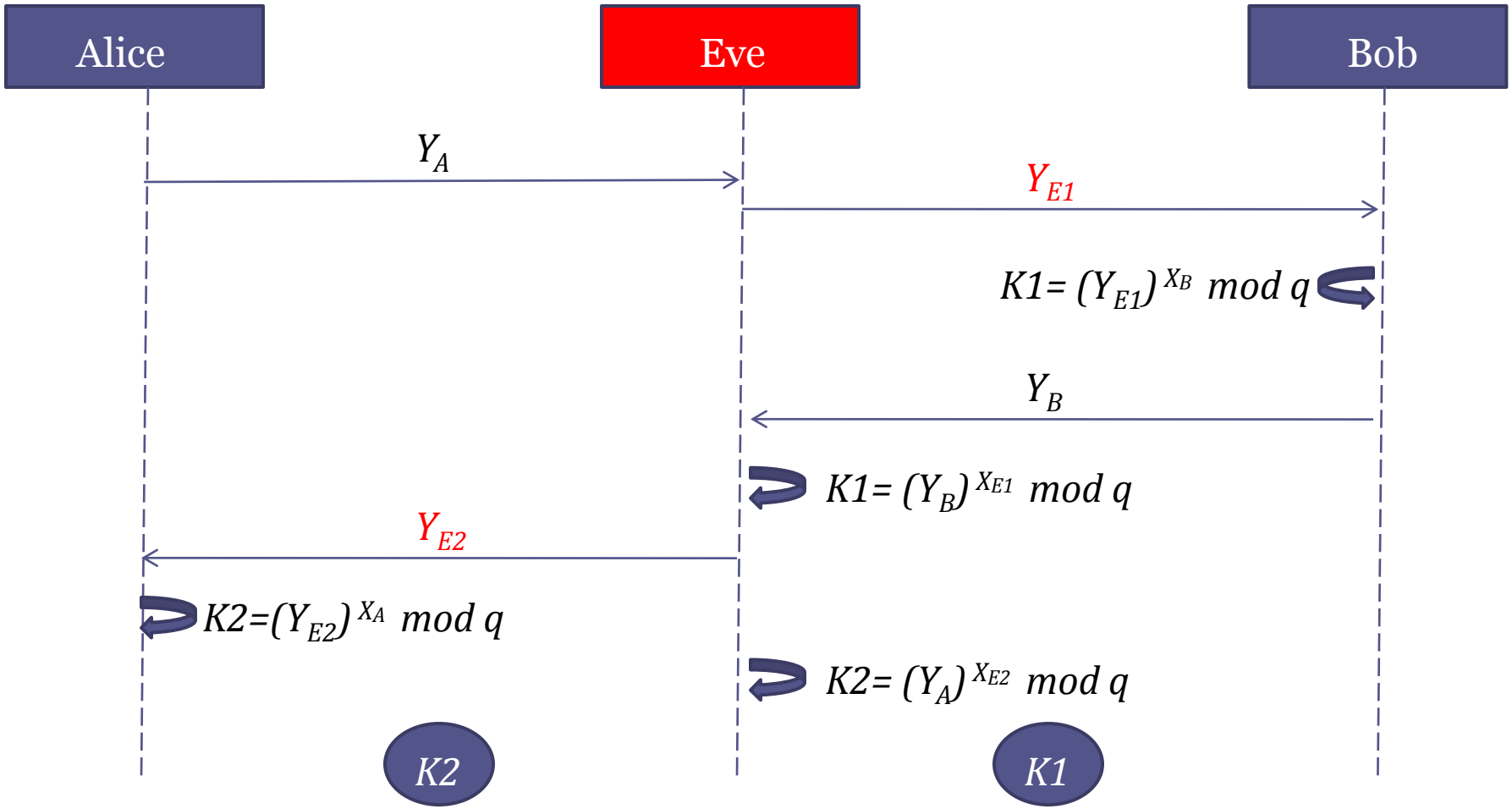
- At this point, *Alice* and *Bob* think that they share a secret key, but instead *Bob* and *Eve* share secret key *K1* and *Alice* and *Eve* share secret key *K2*

Diffie-Hellman Key Exchange Algorithm

- Man-in-the-middle Attack:
- All future communication between *Alice* and *Bob* is compromised in the following way:
  - *Alice* sends encrypted message *M: E(K2,M)*
  - *Eve* intercepts the encrypted message and decrypts it to recover *M*
  - *Eve* sends *Bob E(K1, M)* or *E(K1, M')*, where *M'* is any other message.

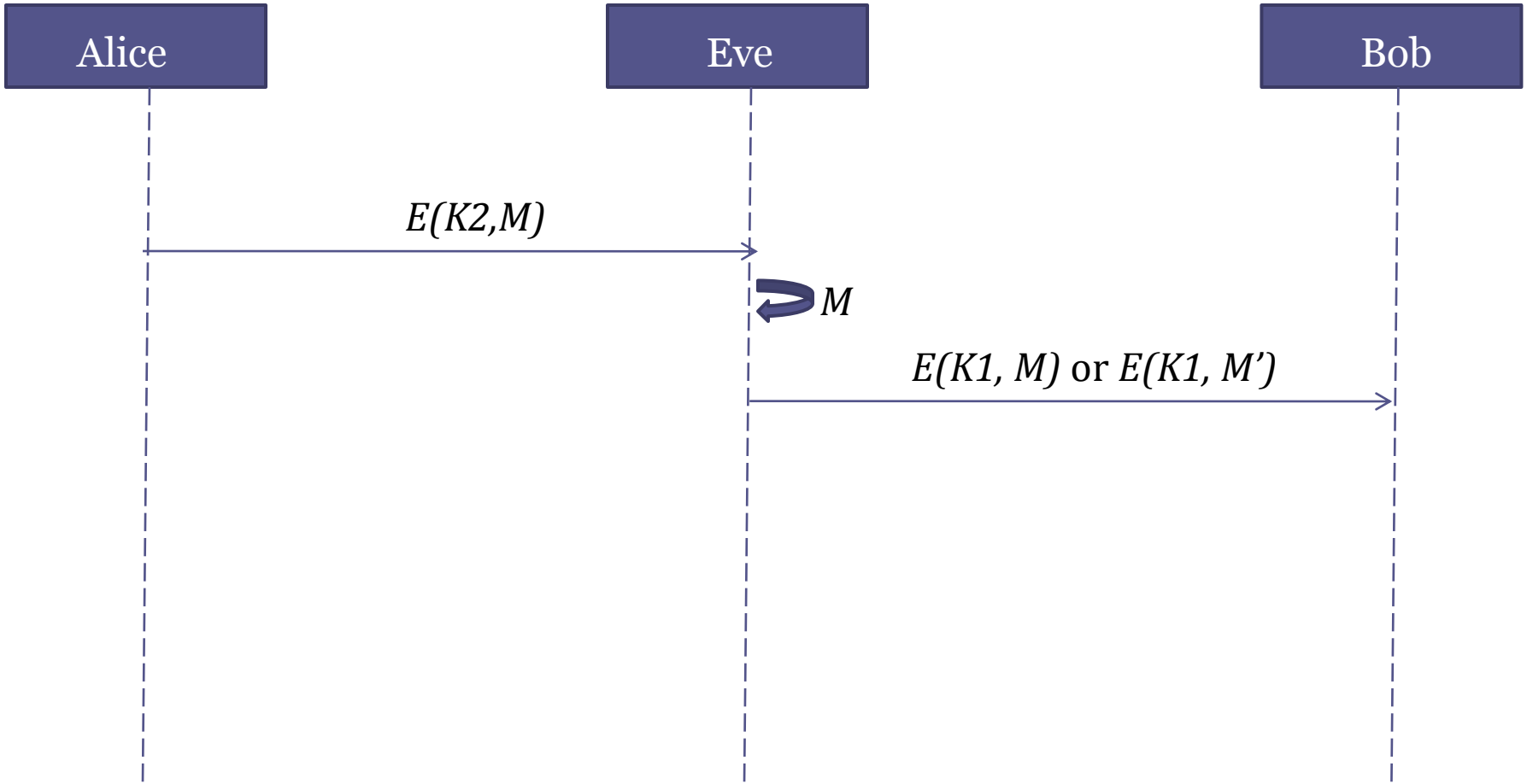
Diffie-Hellman Key Exchange Algorithm:

Man-in-the-middle attack



Diffie-Hellman Key Exchange Algorithm:

Man-in-the-middle attack



END