गृह मंत्रालय
MINISTRY OF
HOME AFFAIRS
सत्यमेव जयते

राष्ट्रीय न्यायिक विज्ञान विश्वविद्यालय
National Forensic Sciences University

NFSU
विद्या अमृतं अश्नुते

D-H Key Exchange and ECC

Dr. Lokesh Chouhan

Associate Professor

गृह मंत्रालय
MINISTRY OF
HOME AFFAIRS
सत्यमेव जयते

राष्ट्रीय न्यायालयिक विज्ञान विश्वविद्यालय
(राष्ट्रीय महत्त्व का संस्थान, गृह मंत्रालय, भारत सरकार)
National Forensic Sciences University
(An Institution of National Importance under Ministry of Home Affairs, Government of India)

NFSU
विद्या अमृतं अश्नुते

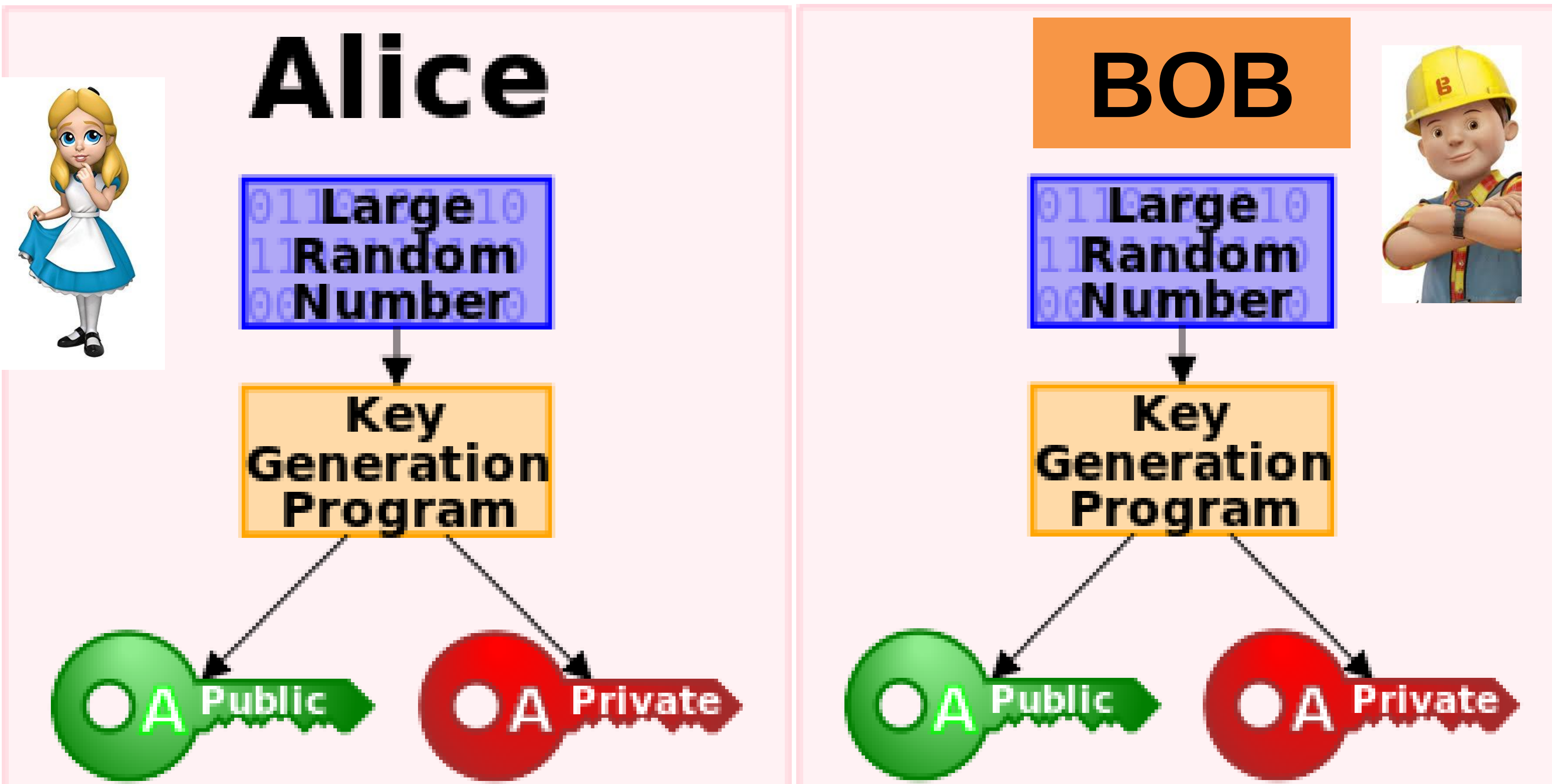
E-Mail: Lokeshchouhan@gmail.com, Lokesh.chouhan_goa@nfsu.ac.in

Mob: +91-898924399, 9827235155

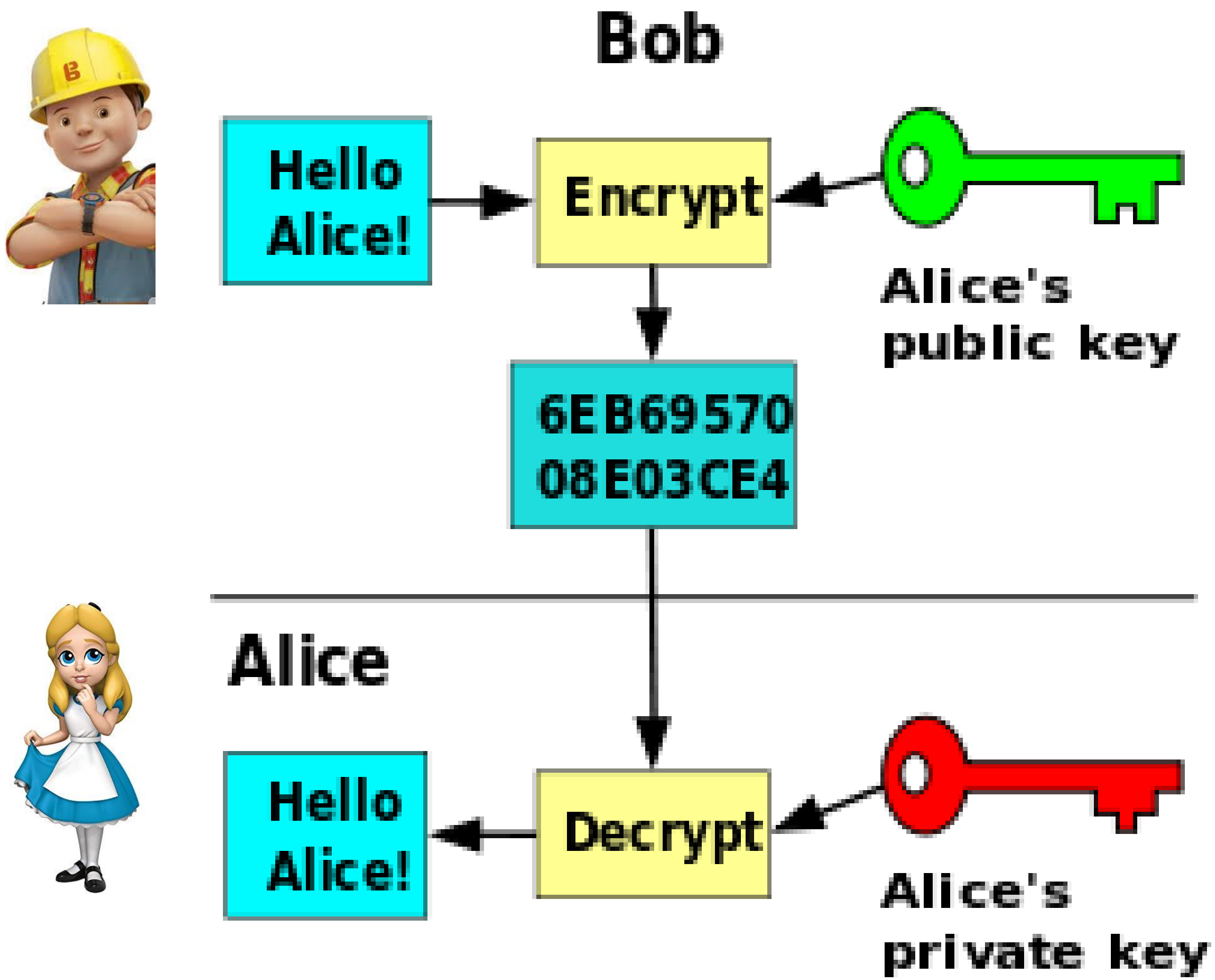
Unit 4

Public Key Cryptography-III

Public Key Cryptosystem (PKCS)



Encryption / Decryption



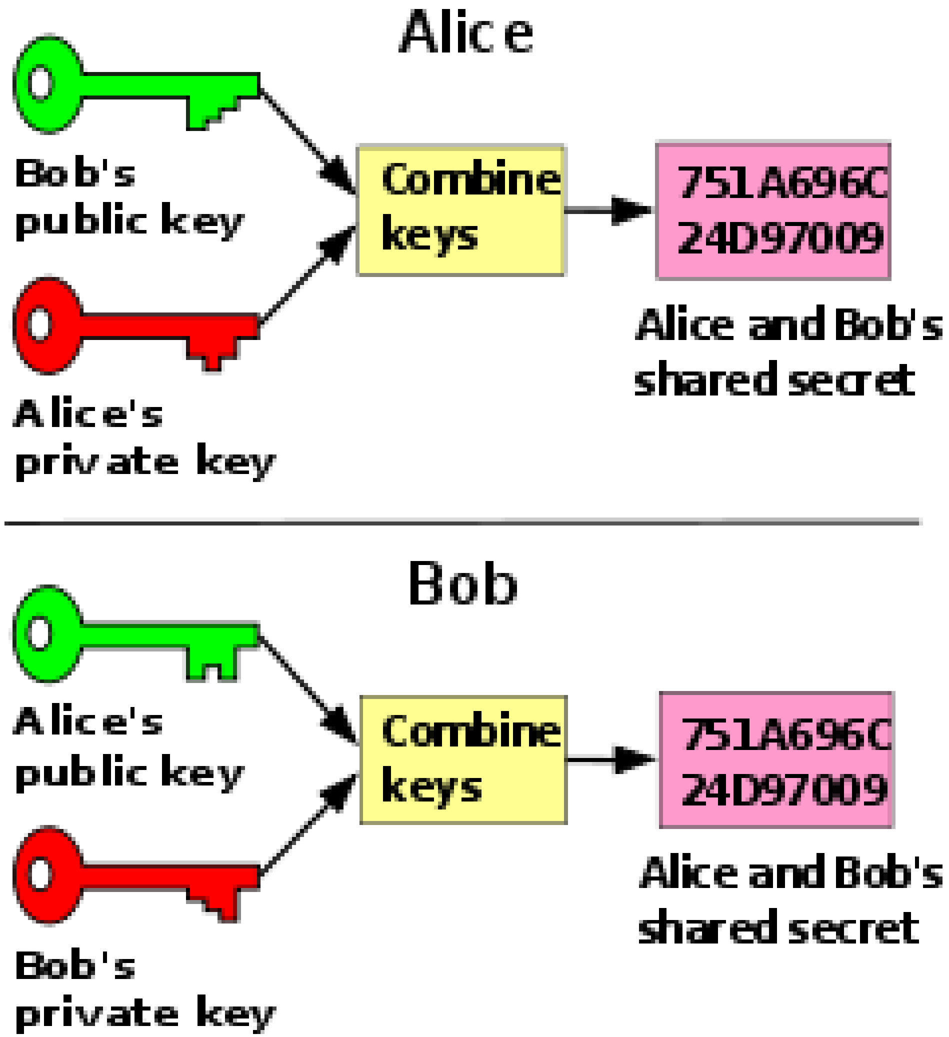
Diffie-Hellman Key Exchange

- First public-key type scheme proposed
- by **Diffie & Hellman** in 1976 along with the exposition of public key concepts
  - note: now know that **Williamson (UK CESG)** secretly proposed the concept in **1970**
- is a practical method for **public exchange of a secret key**
- used in a number of commercial products

Diffie-Hellman Key Exchange

- a public-key distribution scheme
  - **cannot be used to exchange an arbitrary message**
  - rather it can establish a common key
  - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Key Exchange



Diffie-Hellman Setup

- all users agree on **global parameters:**
  1. large **prime integer** or polynomial q
  2. a being a **primitive root mod** q
- each <u>user (eg. A)</u> generates their key
  - chooses a **secret key (number):** $x_A < q$
  - compute their **public key:** $y_A = a^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Setup

- B user will aslo generates their key
  - chooses a **secret key (number):** $x_B < q$
  - compute their **public key:** $y_B = a^{x_B} \bmod q$
- each user makes public that key y_B

Diffie-Hellman Key Exchange

- **Shared session key** for users A & B is K_{AB}:

  - which **B** can compute

    $K_{AB} = y_A^{x_B} \bmod q$

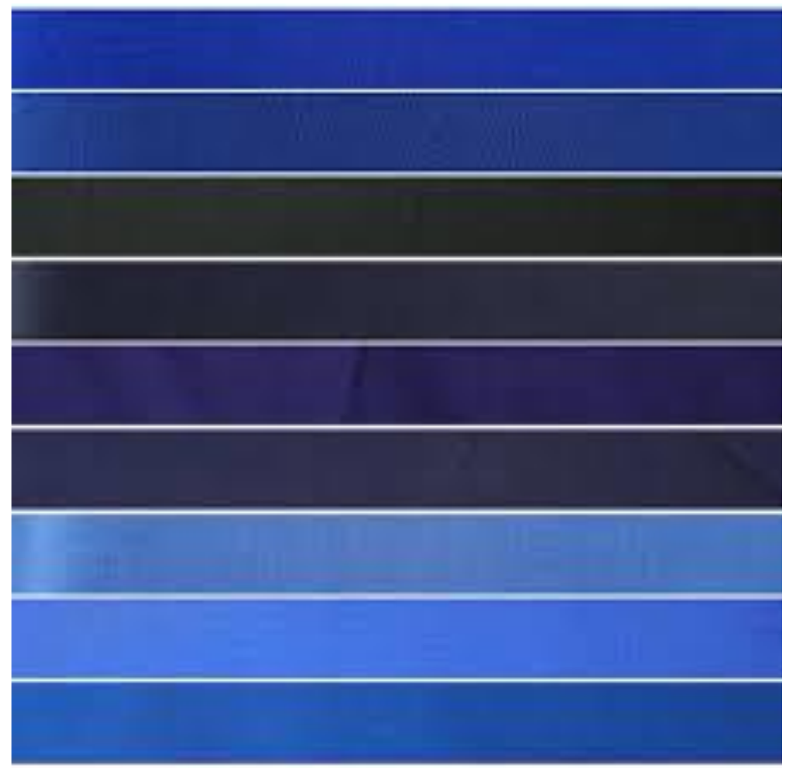
  - which **A** can compute

    $K_{AB} = y_B^{x_A} \bmod q$

- $Z_A = Z_B = K_{AB} = a^{x_A \cdot x_B} \bmod q$
- K_{AB} is used as **session key** in private-key encryption scheme between Alice and Bob
- if **Alice** and **Bob** subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Color Shades

Try to Identify the colors?



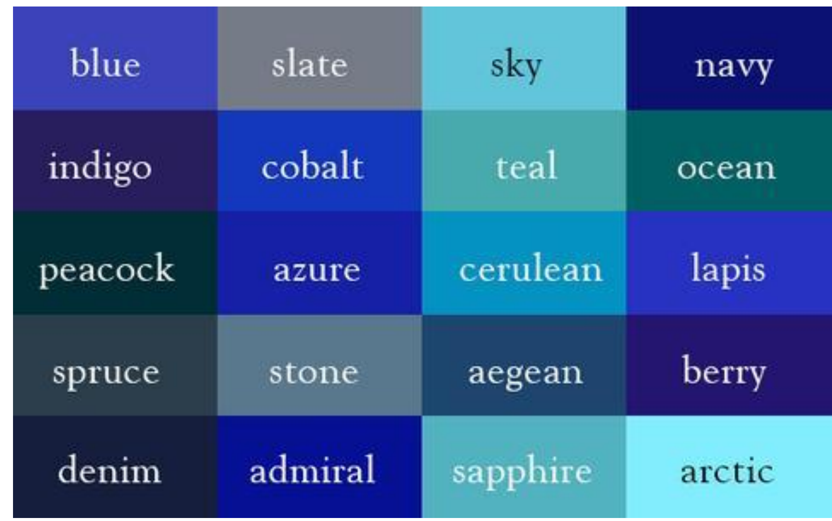
Color Shades

- Pink Blue

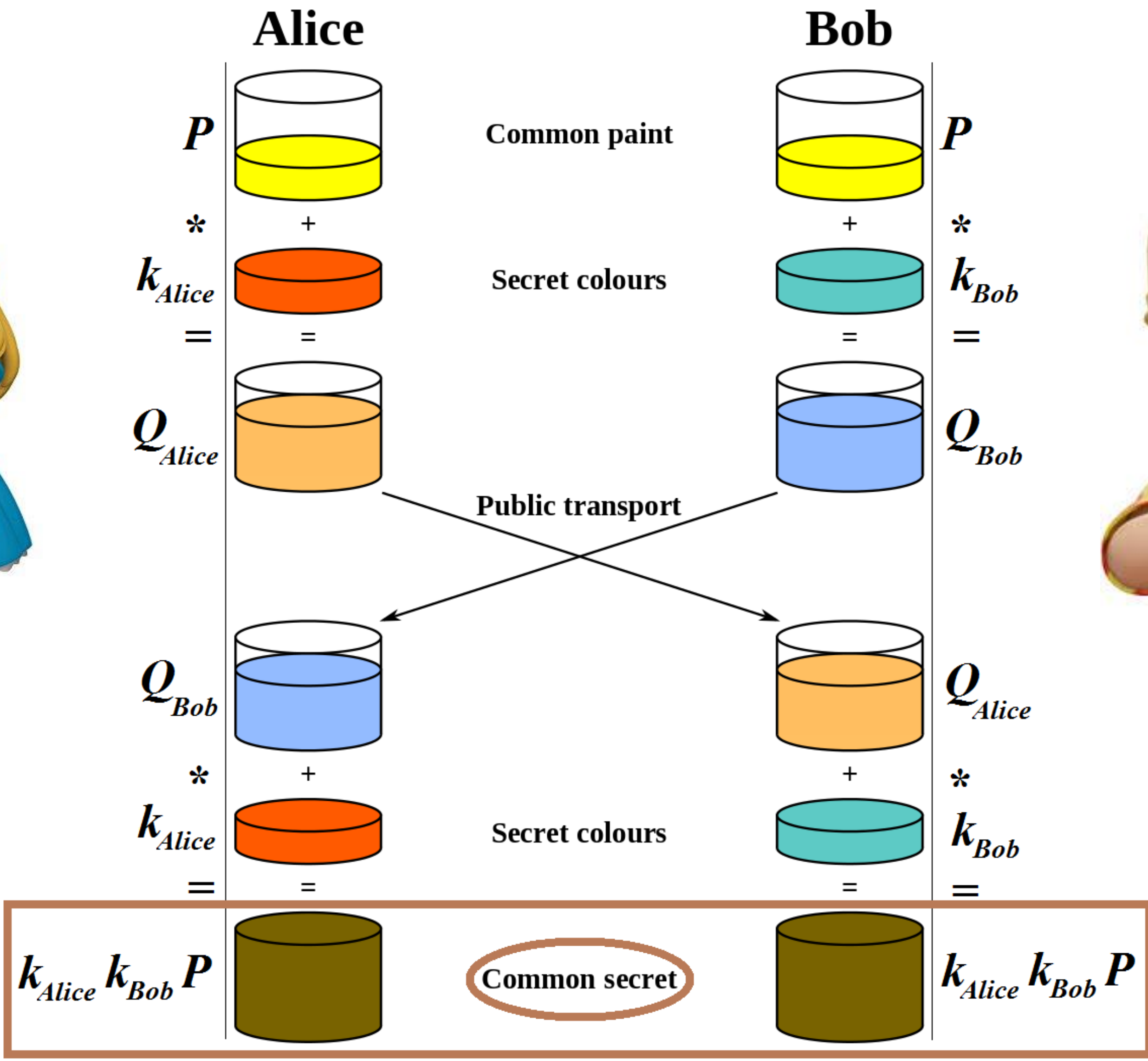
Color Shades

- Pink

Blue



Diffie-Hellman Key Exchange



Diffie-Hellman Example

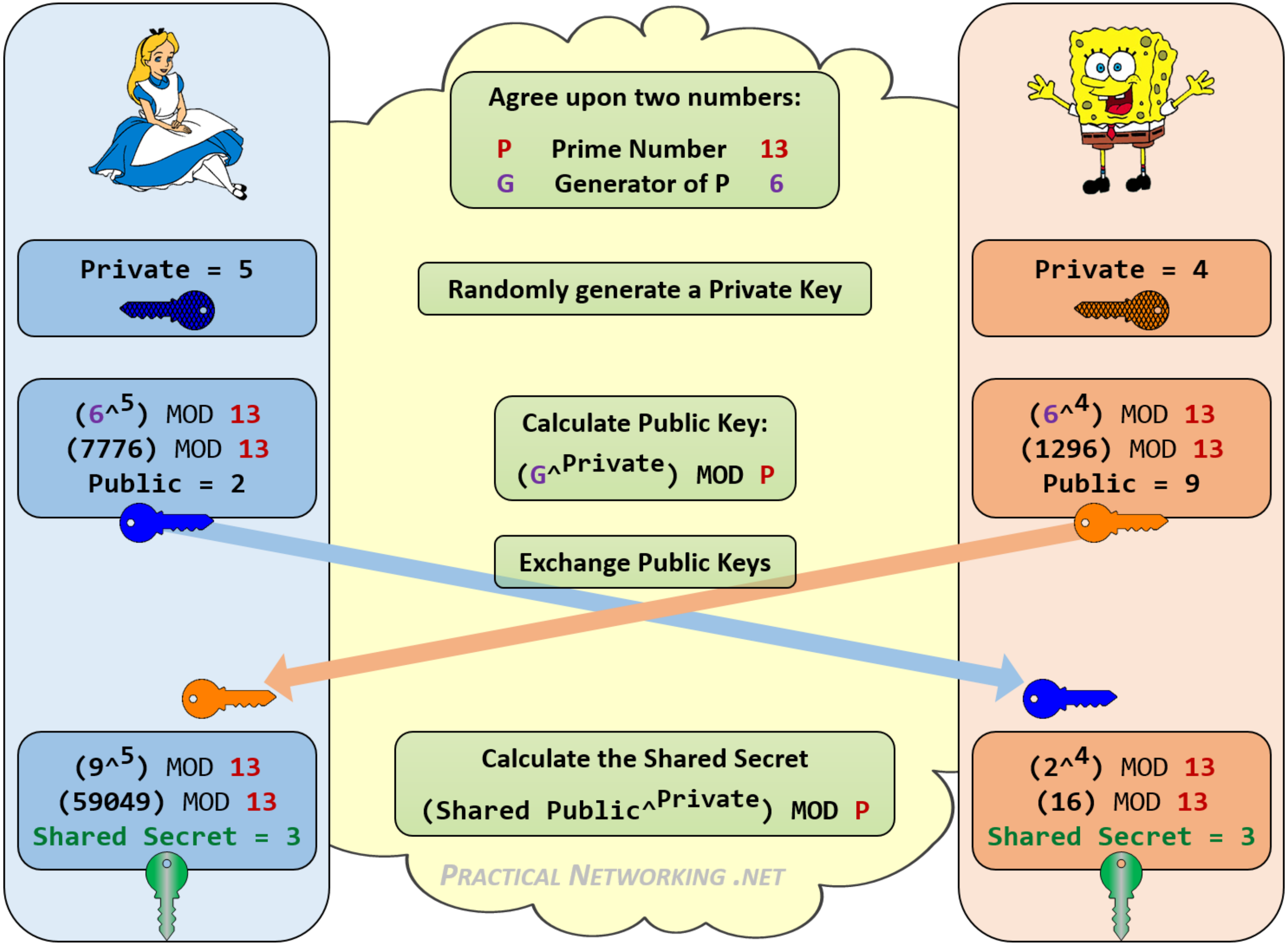
- users **Alice & Bob** who wish to swap keys:
- agree on prime **q=353** and **a=3**
- select random <u>**secret keys**</u>:
  - **A** chooses $x_A=97$,
  - **B** chooses $x_B=233$
- compute respective <u>public keys</u>:
  - $y_A=3^{97} \bmod 353 = 40$ (Alice)
  - $y_B=3^{233} \bmod 353 = 248$ (Bob)
- compute <u>shared session key</u> as:
  - $K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Diffie-Hellman Example

- Global Parameters:
  - q=13 and
  - a=g=6
- $X_A=5$
- $X_B=4$

- Find
  - Public Key
  - Private Key
  - Shared Session Key

Diffie-Hellman Key Exchange

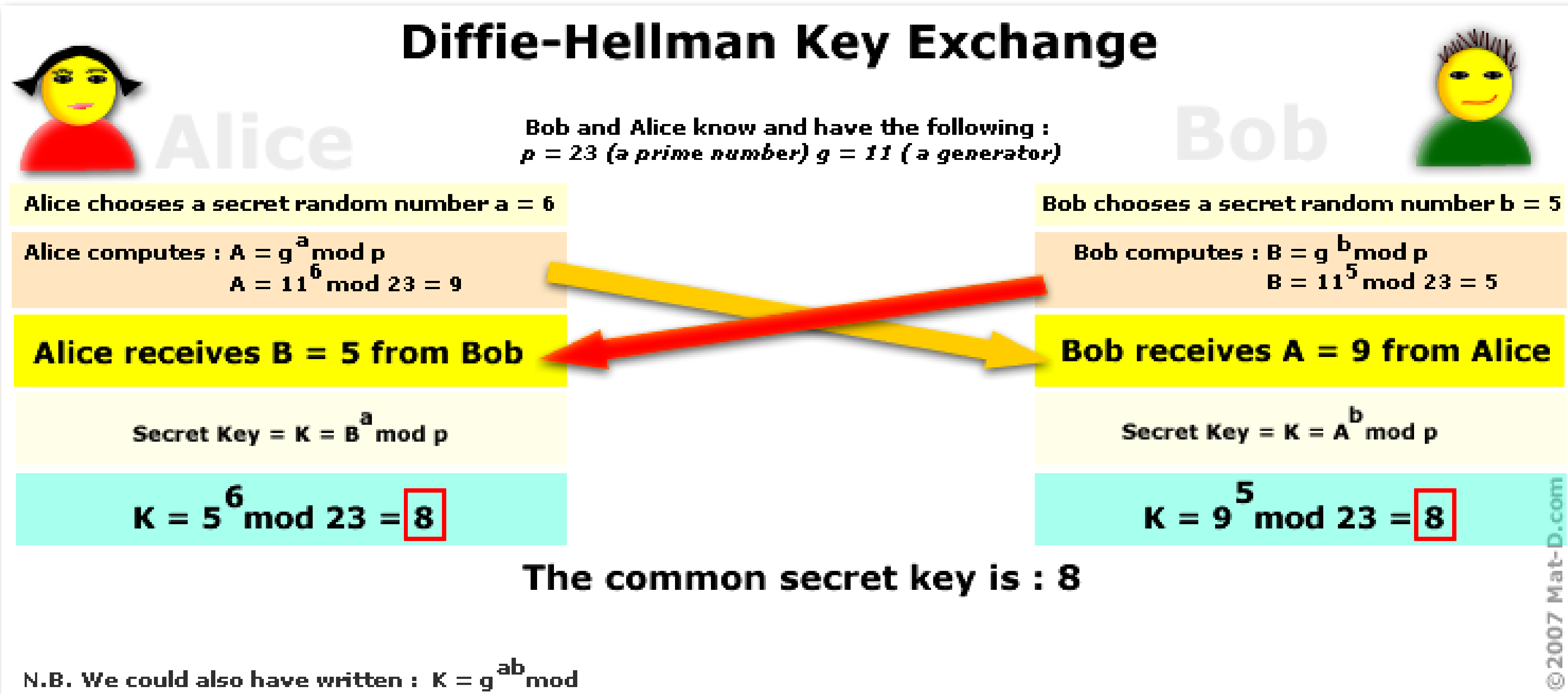


Diffie-Hellman Example

- **Global Parameters:**
  - q=23 and
  - a=g=11
- $X_A=6$
- $X_B=5$

- **Find**
  - Public Key
  - Private Key
  - Shared Session Key

Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange Example

- Que: Calculate Z_A or K_{AB}
- p=11,
- a=2,

  $x_A = 9$,
  $x_B = 4$.

Diffie-Hellman Key Exchange Ex 1

- Here p=11, g=2, $x_A = 9$, $x_B = 4$. So $y_A = 2^{x_A} = 2^9$ (mod 11).
- You can find this most easily by finding 2^2 =4, $2^4 = 4^2 = 16$ (mod 11), $2^8 = (2^4)^2 \;\; 5^2 = 25 \;\; 3$ (mod 11), and finally $2^9 = 2 \times 2^8 \;\; 2 \times 3 = 6$.
- So $y_A = 6$.
- Similary, $2^{x_B} = 2^4 = 16 \;\; 5$ (mod 11), so $y_B = 5$.
- The secret shared key z_A is the remainder of $y_B{}^{x_A} = 5^9$ (mod 11).
- So find $5^2 = 25 \;\; 3$ (mod 11), $5^4 = (5^2)^2 \;\; 3^2 = 9$ (mod 11), $5^8 = (5^4)^2 \;\; 9^2 = 81 \;\; 4$ (mod 11), $5^9 = 5 \times 5^8 \;\; 5 \times 4 = 20 \;\; 9$ (mod 11).
- As a check, z_B is the remainder of $y_A{}^{x_B} = 6^4$ (mod 11).
- $6^2 = 36 \;\; 3$ (mod 11) so $6^4 = (6^2)^2 \;\; 3^2 = 9$ (mod 11), which checks.
- So $z_A = z_B = 9$.

Key Exchange Protocols

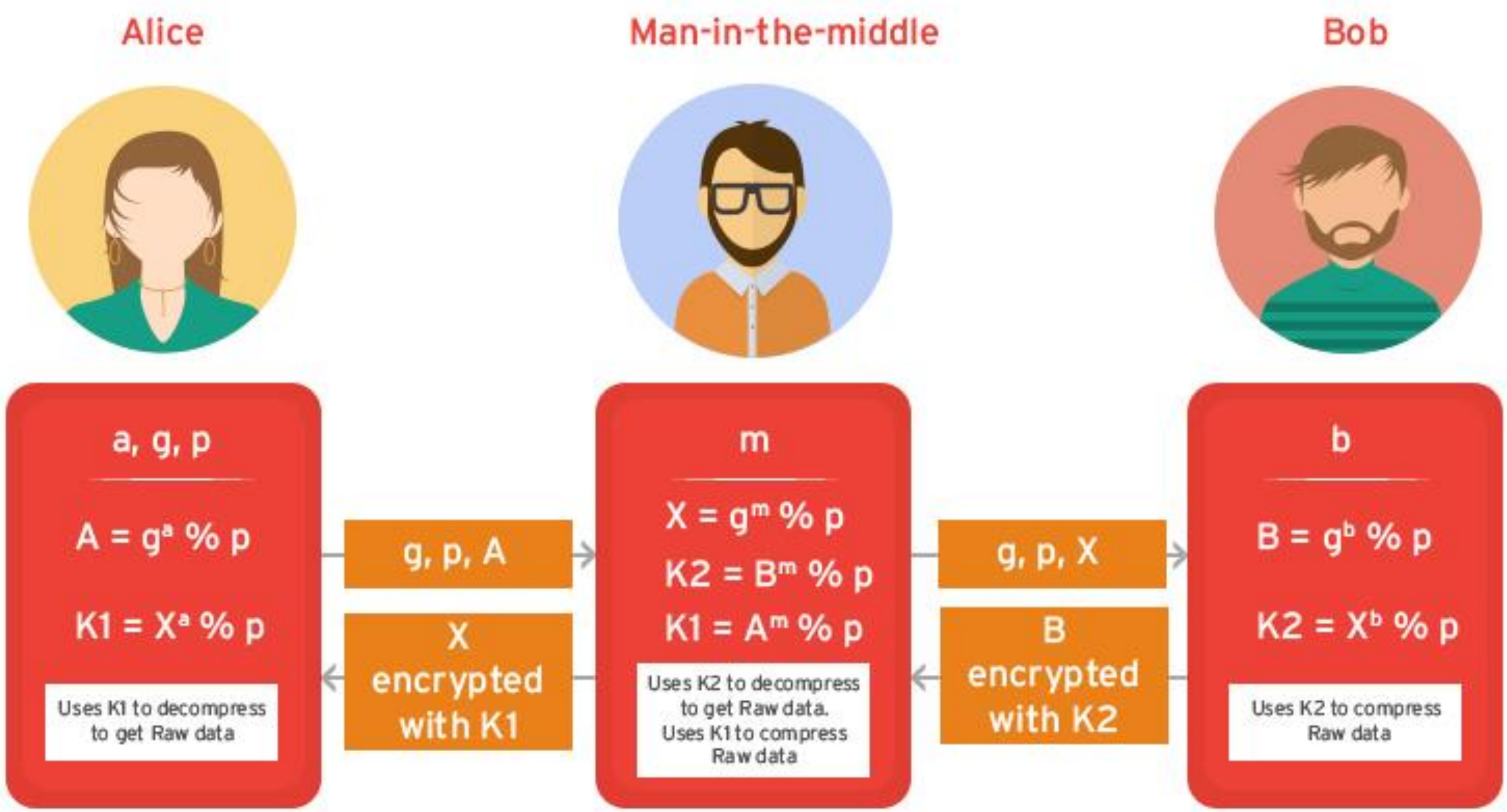
- users could create random private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a meet-in-the-Middle Attack
- authentication of the keys is needed

Man-in-the-Middle Attack

1. Darth prepares by creating two private / public keys
2. Alice transmits her public key to Bob
3. Darth intercepts this and transmits his first public key to Bob. Darth also calculates a **shared key** with Alice
4. Bob receives the public key and calculates the **shared key** (with Darth instead of Alice)
5. Bob transmits his public key to Alice
6. Darth intercepts this and transmits his second public key to Alice. Darth calculates a shared key with Bob
7. Alice receives the key and calculates the shared key (with Darth instead of Bob)

- Darth can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob

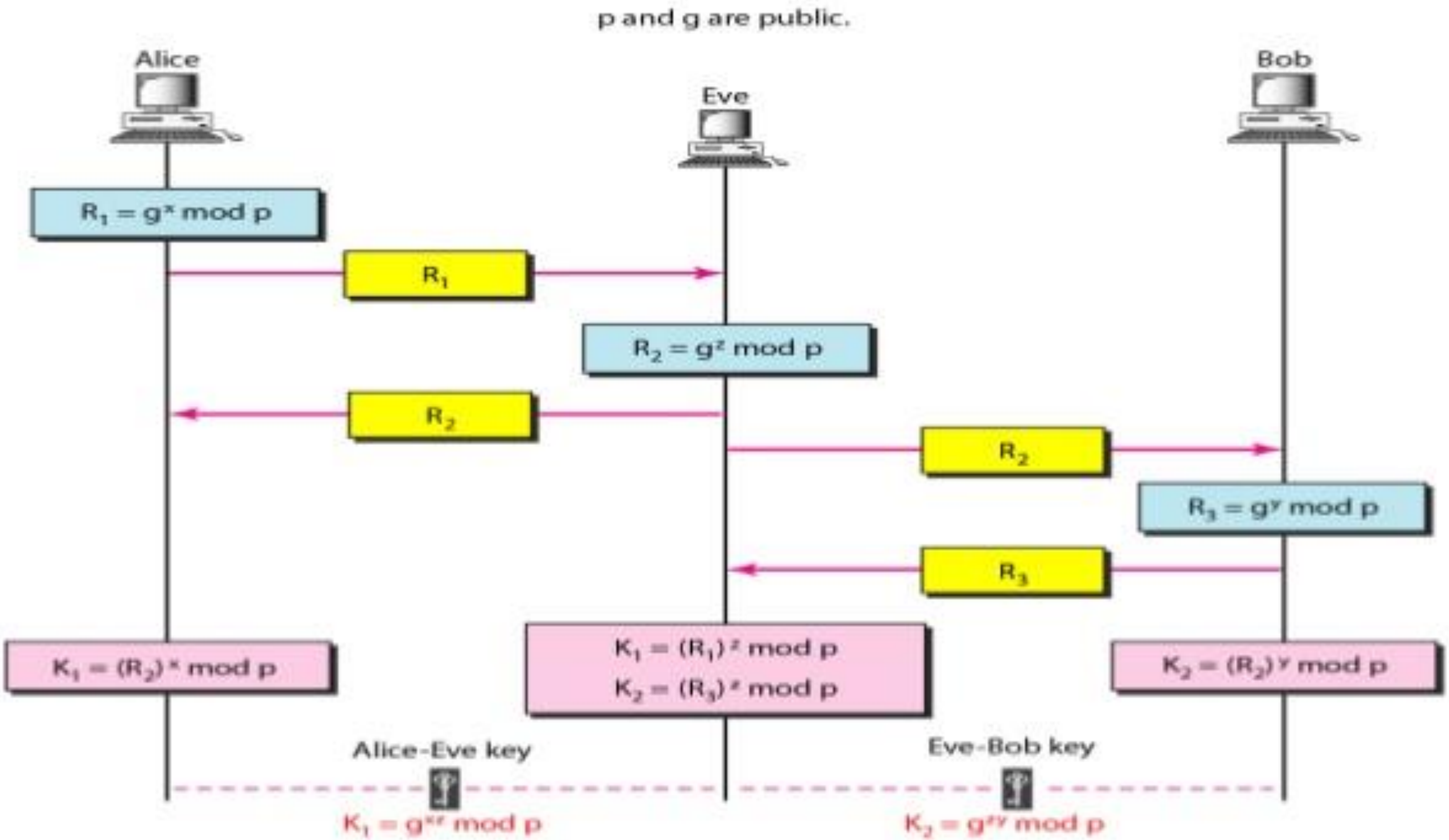
Man-in-the-Middle Attack



$$K1 = A^m \% p = X^a \% p$$

$$K2 = X^b \% p = B^m \% p$$

Man-in-the-middle attack-



Man-in-the-Middle Attack Example

Alice	Mr. X	Bob
$g^{S_A} = 8389$	$g^{S_X} = 5876$	$g^{S_B} = 9267$

$8389 \longrightarrow$ Mr. X; $5876 \longrightarrow$ Bob

Alice $\longleftarrow 5876$; Mr. X $\longleftarrow 9267$

shared key K_{AX}

$$5876^{S_A} = 8389^{S_X}$$

shared key K_{BX}

$$9267^{S_X} = 5876^{S_B}$$

Man-in-the-Middle Attack Example

Alice

Known	Unknown
$p = 23$	b
$g = 5$	
$a = 6$	
$A = 5^a \bmod 23$	
$A = 5^6 \bmod 23 = 8$	
$B = 19$	
$s = B^a \bmod 23$	
$s = 19^6 \bmod 23 = 2$	
$s = 2$	

Bob

Known	Unknown
$p = 23$	a
$g = 5$	
$b = 15$	
$B = 5^b \bmod 23$	
$B = 5^{15} \bmod 23 = 19$	
$A = 8$	
$s = A^b \bmod 23$	
$s = 8^{15} \bmod 23 = 2$	
$s = 2$	

Eve

Known	Unknown
$p = 23$	a
$g = 5$	b
	s
$A = 8$	
$B = 19$	
$s = 19^a \bmod 23 = 8^b \bmod 23$	

Elliptic Curve Cryptography

- majority of public-key crypto (RSA, D-H) use either **integer or polynomial arithmetic** with **very large numbers/polynomials**
- imposes a **significant load** in storing and processing keys and messages
- an alternative is to **use elliptic curves**
- offers same security with smaller bit sizes
- newer, but not as well analysed

Real Elliptic Curves

- an **elliptic curve** is defined by an equation in **two variables x & y**, with **coefficients**
- variables and coefficients are restricted to elements in a **finite field** (in **Cryptography**)
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define **Zero Point O**
- consider set of points E(a,b) that satisfy
- have addition operation for elliptic curve
  - geometrically sum of **P+Q** is **reflection** of the **intersection R**

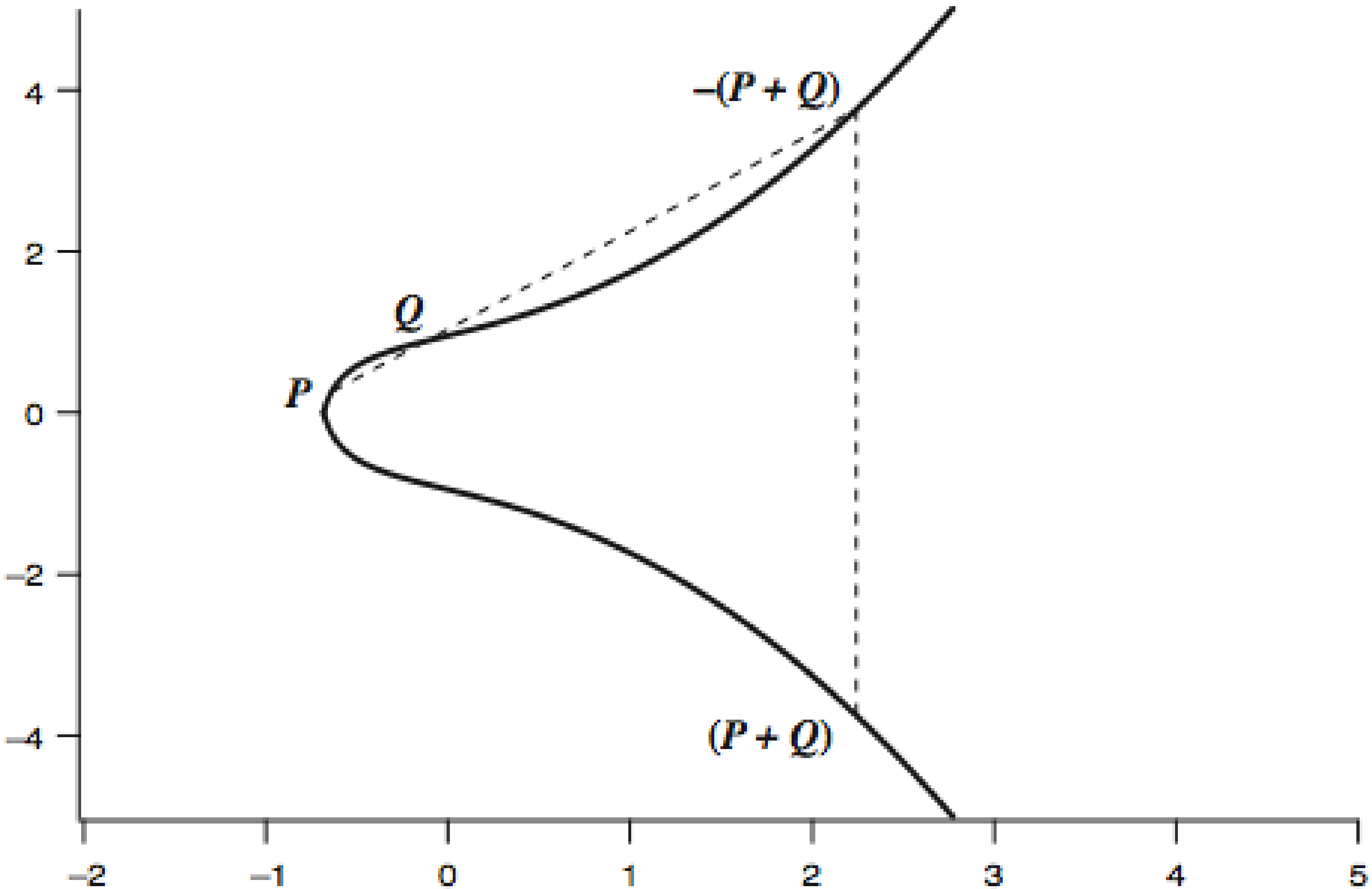
Real Elliptic Curves

- Elliptic curves are not ellipses. They are so named because they are described by cubic equations, similar to those used for calculating the circumference of an ellipse.
- For our purpose, we can consider cubic equations for elliptic curves of the form shown here.
- An elliptic curve is a single element **denoted *O*** and called the ***point at infinity* or the *zero point.***

Real Elliptic Curves

- Now, consider the set of points E(*a*, *b*) consisting of all of the points *(x, y)* that satisfy this equation together with the element ***O***.
- Using a different value of the pair *(a, b)* results in a different set *E(a, b).*
- Can derive an algebraic interpretation of addition, based on computing gradient of tangent and then solving for intersection with curve. There is also an algebraic description of additions over elliptic curves (refer Book).

Real Elliptic Curve Example



(b) $y^2 = x^3 + x + 1$

Real Elliptic Curve Example

- If **three points** on an elliptic curve lie on a straight line, their **sum is *O***. hence define addition as:

1. **O** serves as the additive identity. Thus O = –O; for any point P on the elliptic curve, P + O = P. In what follows, we assume P <> O and Q <> O.
2. The **negative of a point** P is the point with the same x coordinate but the negative of the y coordinate; that is, if P = (x, y), then –P = (x, –y). These two points can be joined by a vertical line & that P + (–P) = P – P = O.

Real Elliptic Curve Example

3. To add two points **P and Q** with **different x coordinates**, draw a straight line between them and find the **third point of intersection R**.

4. There is a unique point **R** that is the point of intersection (unless the line is tangent to the curve at either P or Q, in which case we take R = P or R = Q, respectively).

5. To form a group structure, we need to define addition on these three points as follows:

   P + Q = –R. ie. P + Q to be the mirror image (with respect to the x axis) of the third point of intersection as shown on slide.

Real Elliptic Curve Example

6. The geometric interpretation of the preceding item also applies to two points, P and –P, with the same x coordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point. We therefore have P + (–P) = O, consistent with item (**2**).

7. To double a point Q, draw the tangent line and find the other point of intersection S.
   Then Q + Q = 2Q = –S.

- With the preceding list of rules, it can be shown that the set E(*a*, *b*) is an **abelian group.**

Finite Elliptic Curves

- **Elliptic curve cryptography (ECC)** uses curves whose variables & coefficients are finite
- have **two families** commonly used:
  1. **prime curves $E_p(a,b)$** defined over Z_p
     - use **integers modulo a prime**
     - best in **software**
  2. **binary curves $E_{2^m}(a,b)$** defined over $GF(2^n)$
     - use polynomials with b**inary coefficients**
     - best in **hardware**

Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation
- need **"hard" problem equiv** to **discrete log**
  - **Q=kP,** where **Q,P** belong to a prime curve
  - is "easy" to compute **Q** given **k,P**
  - but **"hard"** to find **k** given **Q,P**
  - known as the elliptic curve logarithm problem
- Certicom example: $E_{23}(9,17)$

- Consider the **group $E_{23}(9, 17)$.**
- This is the group defined by the equation

  $$y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23.$$

- What is the discrete logarithm k of Q = (4, 5) to the base P = (16, 5)? The brute-force method is to compute multiples of P until Q is found.
- Thus *P = (16, 5);*
- *2P = (20, 20);*
- *3P = (14, 14);*
- *4P = (19, 20);*
- *5P = (13, 10);*
- *6P = (7, 3);*
- *7P = (8, 7);*
- *8P = (12, 17) ;*
- *9P = (4, 5).*
- Because 9P = (4, 5) = Q, the discrete logarithm Q = (4, 5) to the base P = (16, 5) is k = 9. In a real application, k would be so **large** as to make the **brute-force approach infeasible.**

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_q(a,b)$
- select base point $G=(x_1,y_1)$
  - with large order n s.t. $nG=O$
- A & B select **private keys** $n_A<n$, $n_B<n$
- compute **public keys**: $P_A=n_A G$, $P_B=n_B G$
- compute **shared key**: $K=n_A P_B$, $K=n_B P_A$
  - same since $K=n_A n_B G$
- attacker would need to **find k, hard**

ECC Encryption/Decryption (ElGamal)

- several alternatives, will consider simplest
- must first encode any **message** **M** as a point on the **elliptic curve** P_m
- select suitable curve & **point** **G** as **in D-H**
- each user chooses **private key** $n_A<n$
- and computes **public key** $P_A=n_AG$
- to encrypt P_m : $C_m=\{kG, P_m+kP_b\}$, **k** random
- decrypt C_m compute:

$$P_m+kP_b-n_B(kG) = P_m+k(n_BG)-n_B(kG) = P_m$$

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is “Pollard rho method”
- compared to factoring, can use much smaller key sizes than with **RSA etc**
- for equivalent key lengths computations are **roughly equivalent**
- hence for similar security ECC offers significant computational advantages

Pros and Cons

- **Pros**
  - **Shorter Key Length**
    - **Same level** of security as **RSA** achieved at a much shorter key length
  - **Better Security**
    - Secure because of the **ECDLP**
    - Higher security per **key-bit than RSA**
  - **Higher Performance**
    - Shorter key-length ensures **lesser power requirement** – suitable in wireless sensor applications and low power devices
    - More computation per bit but overall lesser computational expense or complexity due to **lesser number of key bits**

Pros and Cons

- **Cons**
  - **Relatively newer field (not new now a days)**
    - Idea prevails that all the aspects of the topic may not have been explored yet – possibly unknown vulnerabilities
    - Doesn't have widespread usage
  - **Not perfect**
    - Attacks still exist that can solve ECC (112 bit key length has been publicly broken)
    - Well known attacks are the Pollard's Rho attack (complexity $O(\sqrt{n})$), Pohlig's attack, Baby Step,Giant Step etc

Comparable Key Sizes for Equivalent Security

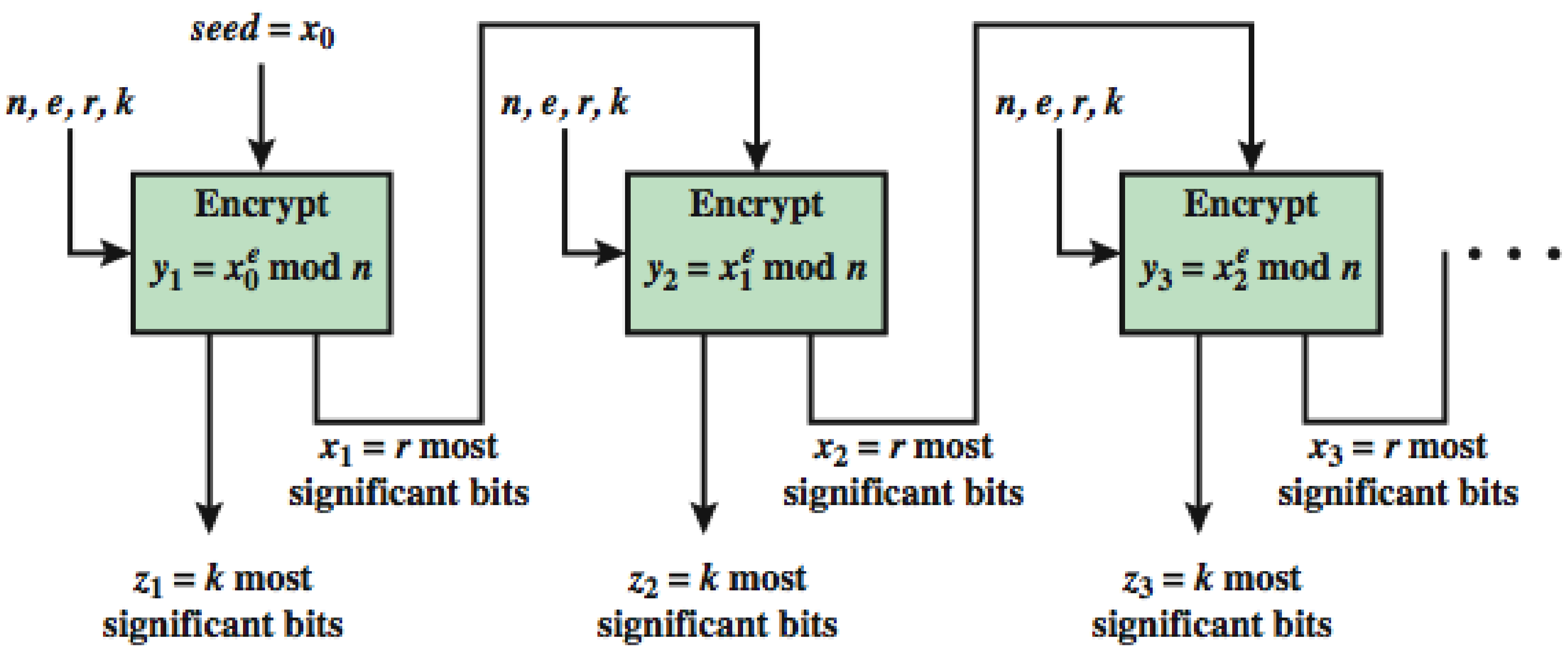
Symmetric scheme (key size in bits)	ECC-based scheme (size of n in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Pseudorandom Number Generation (PRNG) based on Asymmetric Ciphers

- asymmetric encryption algorithm produce apparently random output
- hence can be used to build a pseudorandom number generator (PRNG)
- much slower than symmetric algorithms
- hence only use to generate a short pseudorandom bit sequence (eg. key)

PRNG based on RSA

- have **Micali-Schnorr PRNG using RSA**
  - in ANSI X9.82 and ISO 18031



PRNG based on ECC

- dual elliptic curve PRNG
  - NIST SP 800-9, ANSI X9.82 and ISO 18031
- some controversy on security /inefficiency
- algorithm

```
for i = 1 to k do
set s_i = x(s_{i-1} P )
set r_i  = lsb_240 (x(s_i Q))
end for
return r_1 , . . . , r_k
```

- only use if just have ECC

Summary

- have considered:
  - Diffie-Hellman key exchange
  - ElGamal cryptography
  - Elliptic Curve cryptography
  - Pseudorandom Number Generation (PRNG) based on Asymmetric Ciphers (RSA & ECC)