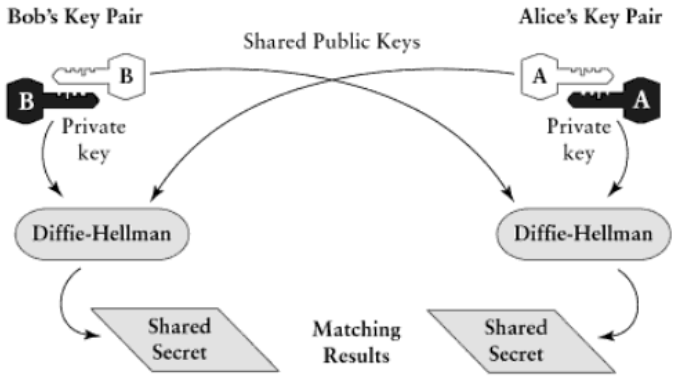
Asymmetric Ciphers: D-H

- **Diffie-Hellman** – first published public-key encryption algorithm (1976)
  - currently used in **TLS** (Transport Layer Security), **SSH**
    IPSec protocol
  - **purpose**: enable two users to securely reach agreement (i.e., generate) a secret key for subsequent symmetric encryption without the involvement of a Key Dist. Cent. (KDC)
  - **property**: private key A and public key B generate the same result as private key B and public key A



Asymmetric Ciphers: D-H (cont.)

- **Diffie-Hellman** – the basics of the math ...

(1) Before establishing a symmetric key, two parties choose/obtain two integer numbers:
p – large prime number with 1024 bits (300 decimal digits)
g – base or generator (primitive root of mod p) – often 2, 3, 7

(2) Alice chooses a large random number **x** ($0 \le x \le p-1$)
and calculates $R_x = g^x \bmod p$.
x: Alice's private key
R_x: Alice's public key

(3) Bob chooses another large random number **y** ($0 \le y \le p-1$)
and calculates $R_y = g^y \bmod p$.
y: Bob's private key
R_y: Bob's public key

(4) Alice sends Bob R_x, Bob sends Alice R_y.

(5) Alice calculates $K = (R_y)^x \bmod p$.

(6) Bob calculates $K = (R_x)^y \bmod p$.

$$K = (g^y \bmod p)^x \bmod p = (g^x \bmod p)^y \bmod p = g^{xy} \bmod p$$

Asymmetric Ciphers: D-H (cont.)

Example: Who knows what ....

Alice		Bob		Eve	
Known	Unknown	Known	Unknown	Known	Unknown
$p = 23$		$p = 23$		$p = 23$	
$g = 5$		$g = 5$		$g = 5$	
$a = 6$	b	$b = 15$	a		a, b
$A = 5^a \bmod 23$		$B = 5^b \bmod 23$			
$A = 5^6 \bmod 23 = 8$		$B = 5^{15} \bmod 23 = 19$			
$B = 19$		$A = 8$		$A = 8, B = 19$	
$s = B^a \bmod 23$		$s = A^b \bmod 23$			
$s = 19^6 \bmod 23 = 2$		$s = 8^{15} \bmod 23 = 2$			s

A's private key

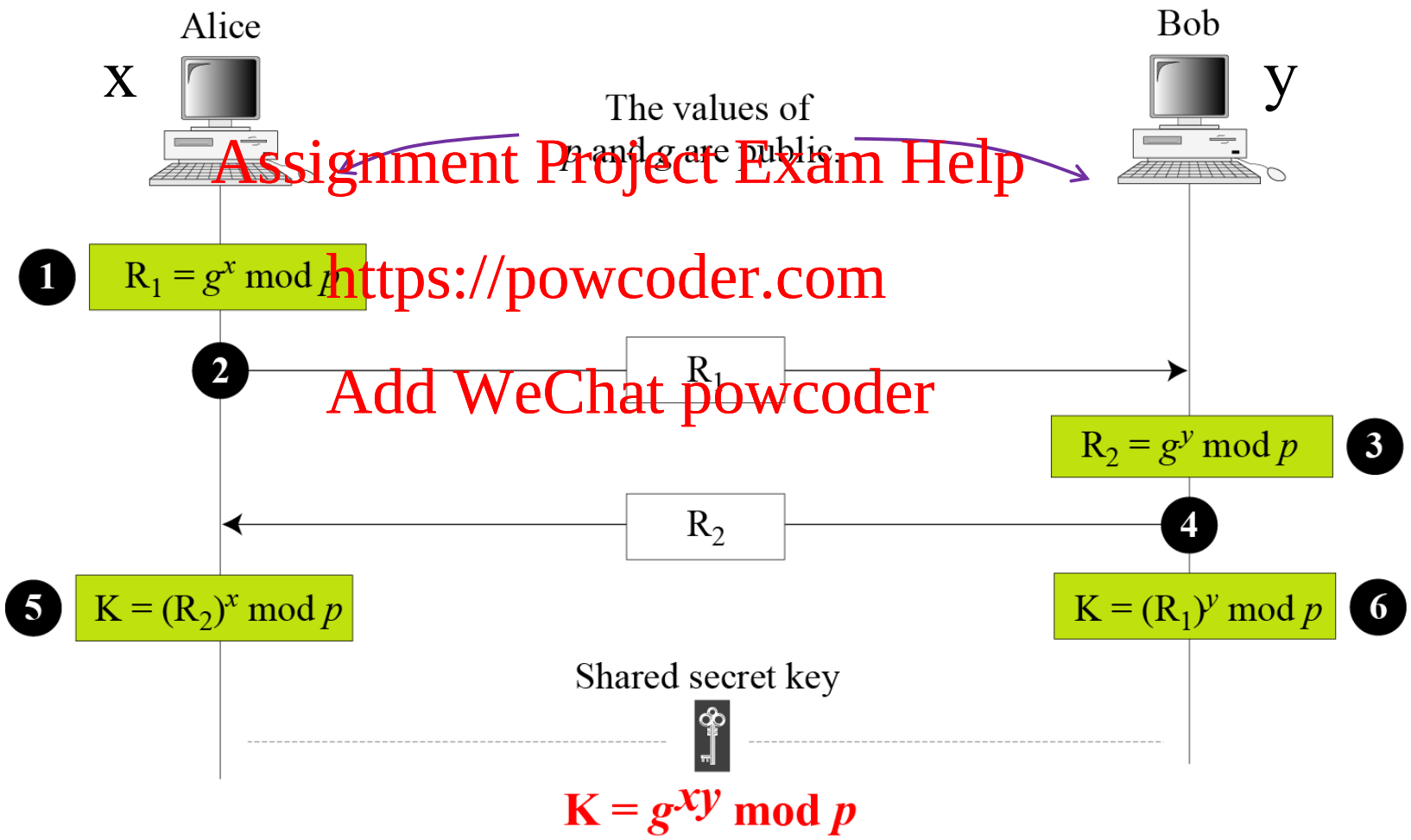
A's public key

B's public key

jointly generated secret symmetric key

Asymmetric Ciphers: D-H (cont.)

Example: Diffie-Hellman exchange



Asymmetric Ciphers: D-H (cont.)

Example: Diffie-Hellman key calculation

Assume that $p = 23$ and $g = 7$.

1. Alice picks $x = 3$ and calculates $R_1 = 7^3 \bmod 23 = 21$.
2. Bob picks $y = 6$ and calculates $R_2 = 7^6 \bmod 23 = 4$.
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates $K = 4^3 \bmod 23 = 64 \bmod 23 = 18$.
6. Bob calculates $K = 21^6 \bmod 23 = 85766121 \bmod 23 =$ $= 18$.
7. The value of K is the same for both Alice and Bob. $g^{xy} \bmod p = 7^{18} \bmod 23 = 18$.

Assume that $p = 7$ and $g = 2$. Alice chooses $a = ??$

Bob

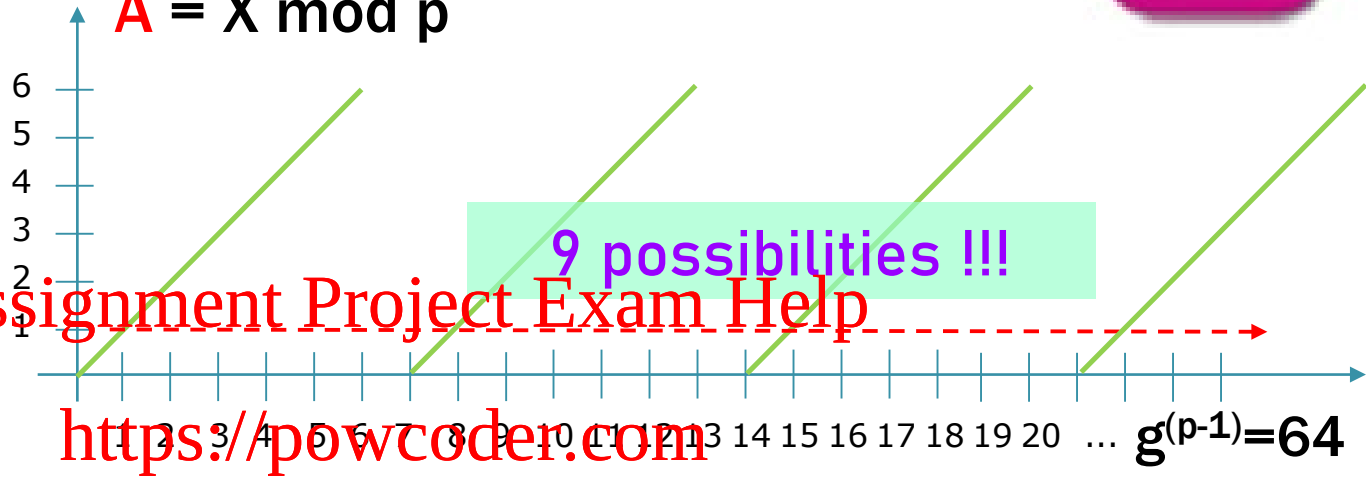
$A = g^{?} \bmod p = 1$ ← $A = g^{a} \bmod p = 1$

Is it really so hard to determine what a is ???

Assume that $p = 7$ and $g = 2$. Alice chooses $a = 3$.

Assume that $p = 7$ and $g = 2$.

$A = X \bmod p$

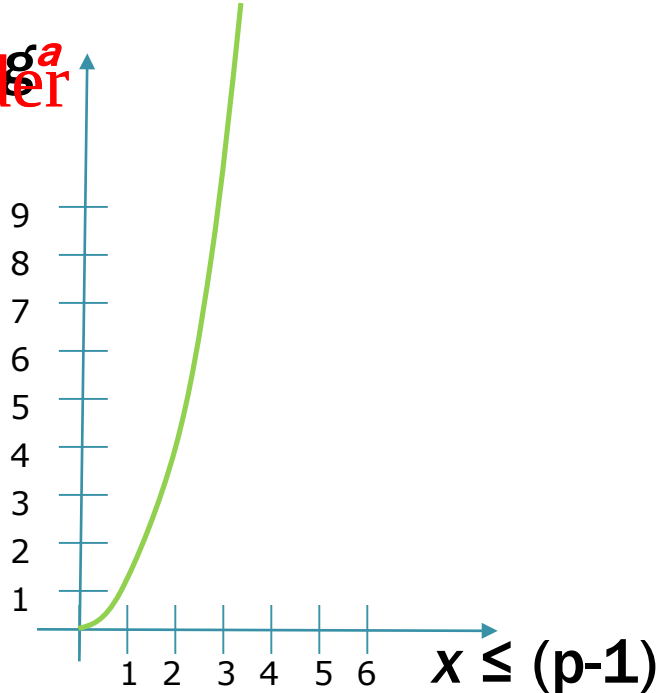


A = 1

9 possibilities !!!

is that number of the form: 2^x

$X = g^a$



Breaking DH algorithm is computationally hard/infeasible for large p.

Asymmetric Ciphers: D-H (cont.)

Example: Diffie-Hellman – more realistic example:
p is 512 bits long

p	76462429856349357218249376595503050747633809672694974892357377286092523566666075542363742330966118003333810619473013095041473870099917804365487858079875 81
g	2
x	557
y	273

R_1	84492028420566550521617294749103509414343369852001266086286363106767361995928082858670080213185929094514021750031997331294583608382194306596602015795535 4
R_2	43526283870920037947074711489558162763638911626211555797512337921856631001143571820839004018187648684175383116534269163026342110672150858962552012885941 43
K	15563800066452229059622582752327076527321804694442367852032040014640650088793665120425742677660832791101715303867456125221315161097658420012040864336177 40

Asymmetric Ciphers: D-H (cont.)

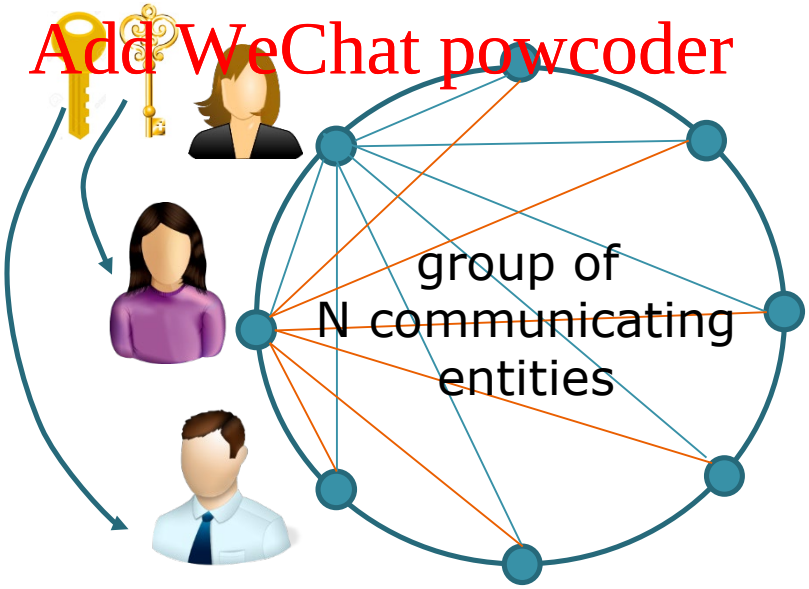
Example: How big is 64-bit int ??

$$9{,}223{,}372{,}036{,}854{,}775{,}807$$

The number 9,223,372,036,854,775,807, equivalent to the hexadecimal value 7FFF,FFFF,FFFF,FFFF16, is the maximum value for a 64-bit signed integer in computing. It is therefore the maximum value for a variable declared as a long integer (`long` , `long long int` , or `bigint`) in many programming languages running on modern computers.[1][2][3]

Asymmetric Ciphers: D-H (cont.)

With DH algorithm
if n people were to securely communicate
$O(n^2)$ message would still
have to be exchanges.



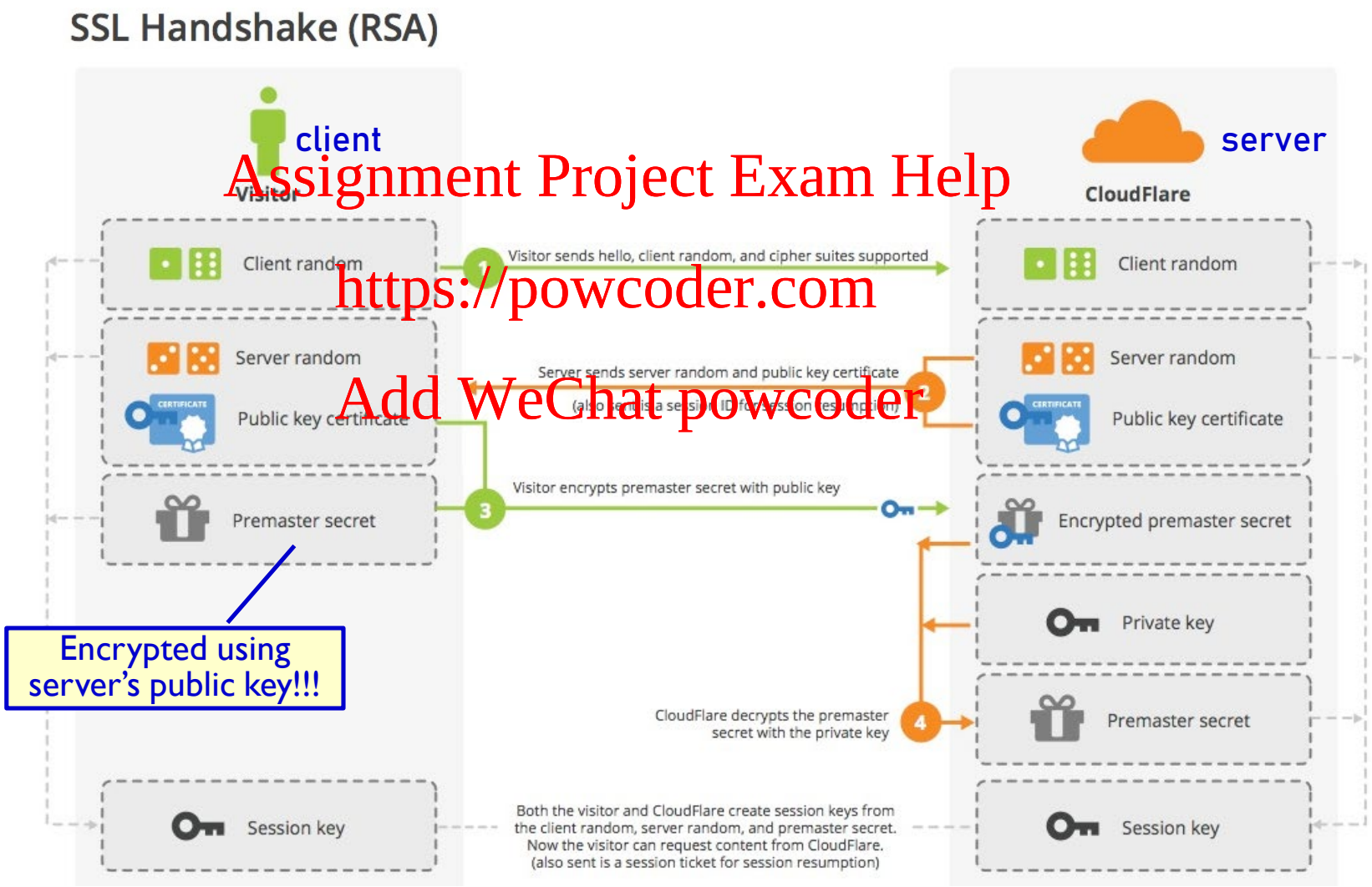
Asymmetric Ciphers: RSA

- **RSA** – Rivest, Shamir, Adleman (1978, MIT)
  - first practically deployable public-key algorithm for secure data transmission and other applications
  - was patented, but patent expired in 2000
  - RSA Security LLC – manufactures security solutions deploying RSA, now owned by Dell …
    - spin-off company: Verisign (1995), bought by Symantec and now DigiCert
  - based on practical difficulty of factoring the product of two large prime numbers
    - like DH uses modulus arithmetic, but in a different way

DH is used to generate a secret key [key agreement] …
RSA is used to exchange a secret key [key transport] …
for subsequent symmetric encryption.

Asymmetric Ciphers: RSA (cont.)

- internet protocols that use RSA: TSL, SSH, IPsec



Asymmetric Ciphers: RSA (cont.)

Example: Excellent video!

https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/intro-to-rsa-encryption

Asymmetric Ciphers: RSA (cont.)

- **RSA** – basics of the math behind key establishment

(1) Choose two **random** large prime numbers **p** and **q**.
The larger the numbers, the more difficult it is to break RSA, but longer it also takes to perform encoding and decoding!!!
RSA Laboratories recommends that the product of **p** and **q** be 1024 bits long.

(2) Compute $n = p \cdot q$ and $z = (p-1) \cdot (q-1)$.

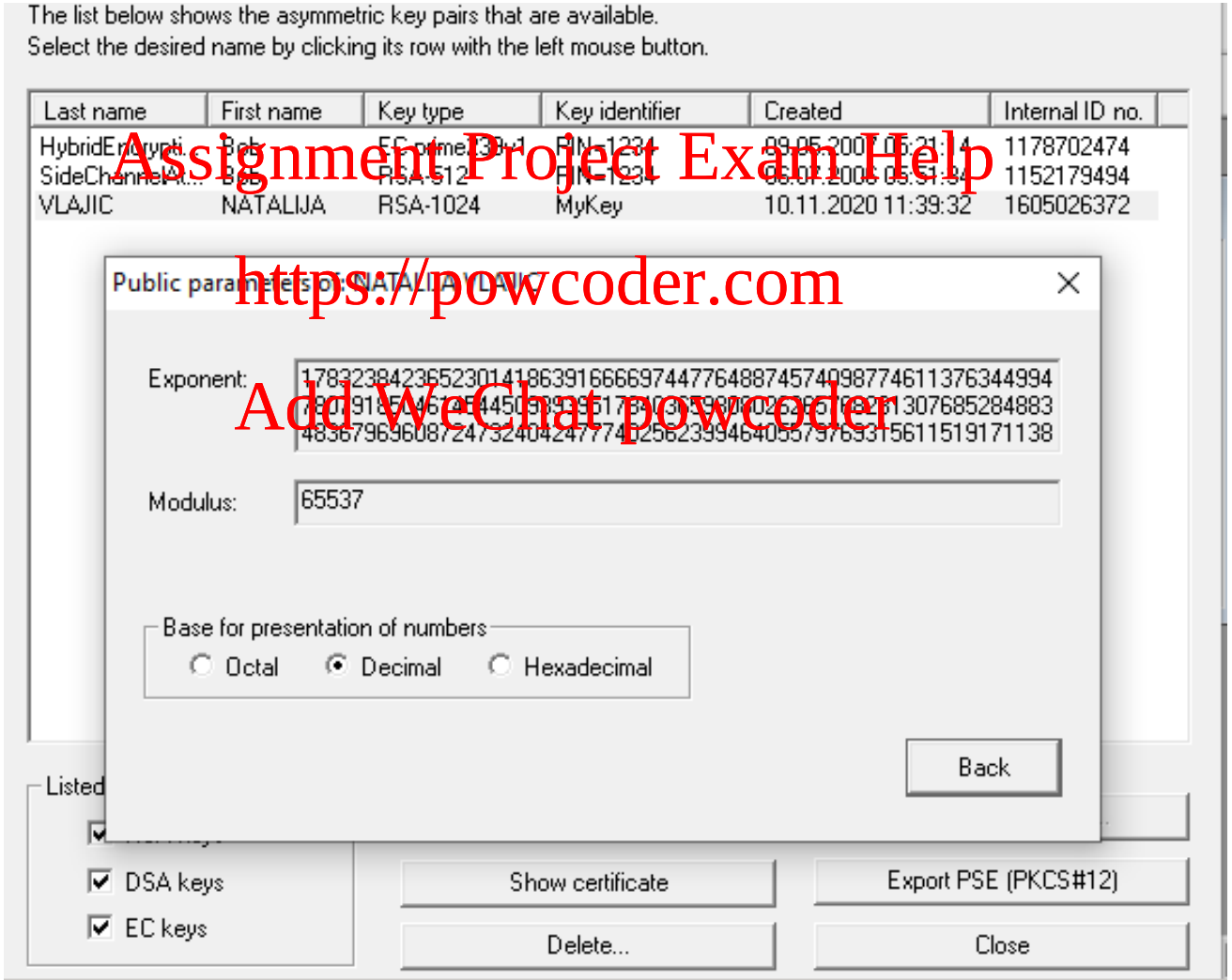
(3) **Choose** a number $e < n$ with no common factors with **z** other than 1. (e – used in encryption, public key.)

(4) **Find** a number **d** such that **ed–1** is exactly divisible by **z**.
That is, choose **d** such that $ed \bmod z = 1$.
(d – used in decryption, private key.)

(5) $K_{public} = (n, e)$, $K_{private} = (n, d)$

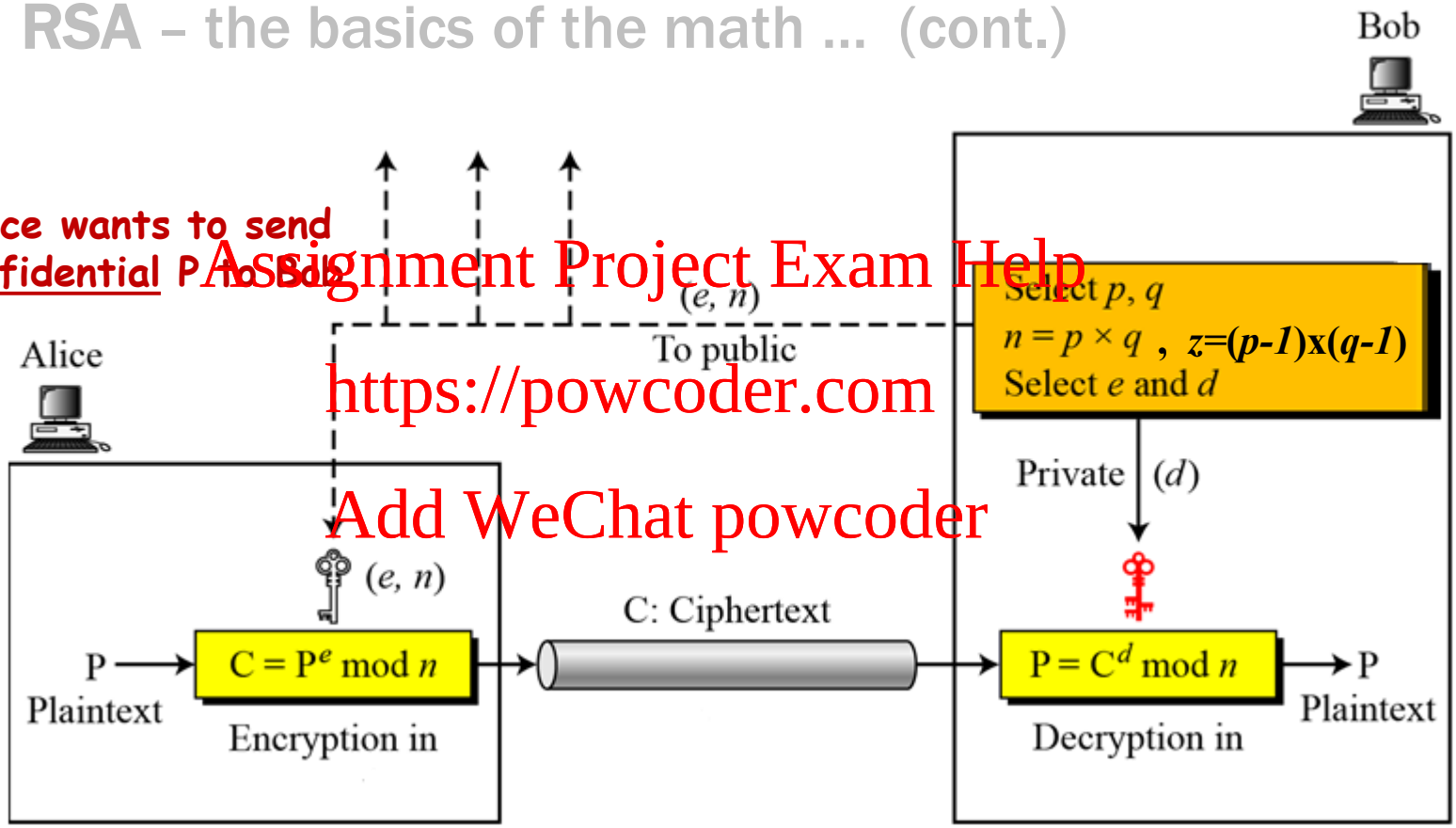
Asymmetric Ciphers: RSA (cont.)

Example: Lab 3 ...



Asymmetric Ciphers: RSA (cont.)

- RSA – the basics of the math … (cont.)



A Encrypts: $C = P^e \bmod n$

B Decrypts: $P = C^d \bmod n = (P^e \bmod n)^d \bmod n$

Asymmetric Ciphers: RSA (cont.)

- **RSA** – the basics of the math …
  - how can we prove:

$$P = (P^e \bmod n)^d \bmod n$$

1) modulo rules allow:

$$= (P^{ed} \bmod n) \bmod n = P^{ed} \bmod n =$$

2) theory of large prime numbers allows:

$$= P^{ed} \bmod n =$$

$$= P \quad - \quad \text{when } P<n$$

proof also holds if **e** and **d** applied in reverse order

Asymmetric Ciphers: RSA (cont.)

- **RSA** – important properties

1) Given $(e, n) = K_{public}$ it is/should be impossible to compute $(d, n) = K_{private}$.

2) The public and private keys are 'commutative'.

$$K_{public}(K_{private}(P)) = K_{private}(K_{public}(P)) = P$$

$$K^{+}(K^{-}(P)) = K^{-}(K^{+}(P)) = P$$

Asymmetric Ciphers: RSA (cont.)

Example: RSA used to encrypt 8-bit messages

Bob chooses **p**=5, **q**=7. Then **n**=35, **z**=24.

e=5 (so **e**, **z** relatively prime).

d=29 (so **ed**-1 exactly divisible by **z**).

Plaintext must be converted to a decimal number!!!

Encrypting 8-bit message: $0000\ 1100_2 = 12_{10}$.

	m	m^e	$c = m^e \bmod n$
Encrypt: (e,n)	12	24832	17

	c	c^d	$m = c^d \bmod n$
Decrypt: (d,n)	17	481968572106750915091411825223071697	12

Asymmetric Ciphers: RSA (cont.)

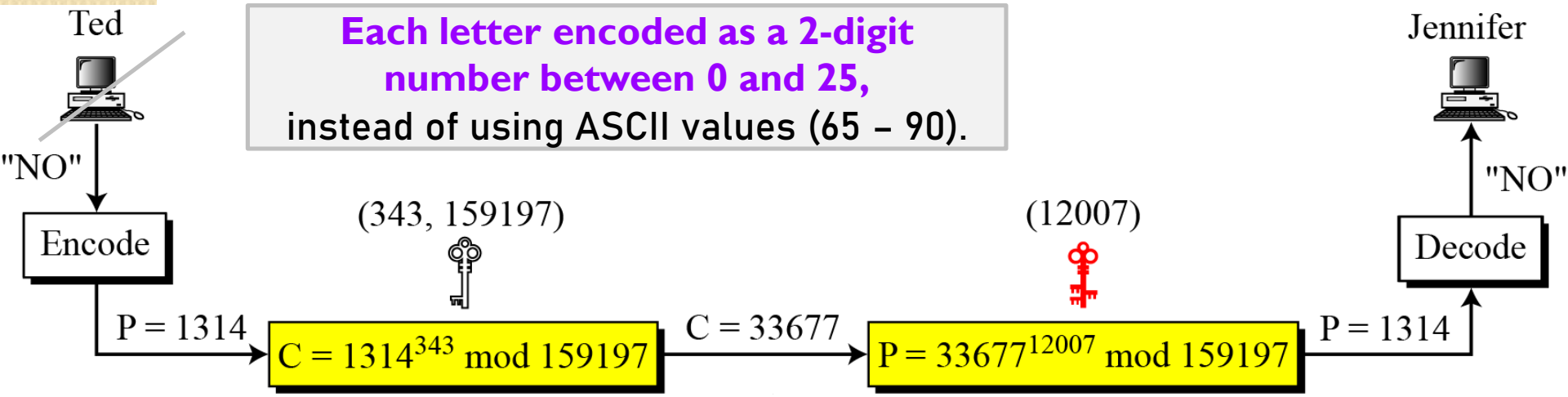
Example: RSA used to encrypt letters

Jennifer creates a pair of keys for herself:

p=397 and ***q***= 401 => **n**=159197 and **z**= 158400.

She then chooses ***e***=343 and **d**=12007.

Show how Ted can send a 2-letter text message to Jennifer if he knows ***e*** and ***n***.



Example: RSA and NSE Saga

Security firm RSA took millions from NSA: report

The National Security Agency paid $10 million to the security firm RSA to implement intentionally flawed encryption, according to a new report.

From 2004 to 2013, RSA shipped security software — BSAFE toolkit and Data Protection Manager — that included a default cryptographically secure pseudorandom number generator, Dual EC DRBG, that was later suspected to contain an alleged secret National Security Agency backdoor.

In 2014, the Snowden leaks revealed how the NSA was effectively infiltrating crypto standards efforts to take control of them and make sure that backdoors or other weaknesses were installed.

On 20 December 2013, Reuters' Joseph Menn reported that NSA secretly paid RSA Security $10 million in 2004 to set Dual_EC_DRBG as the default CSPRNG in BSAFE. The story quoted former RSA Security employees as saying that "no alarms were raised because the deal was handled by business leaders rather than pure technologists".

https://en.wikipedia.org/wiki/RSA_Security

https://www.reuters.com/article/us-usa-security-rsa/exclusive-secret-contract-tied-nsa-and-security-industry-pioneer-idUSBRE9BJ1C220131220

https://www.techdirt.com/articles/20131220/14143625655/nsa-gave-rsa-10-million-to-promote-crypto-it-had-purposely-weakened.shtml

http://news.cnet.com/8301-1009_3-57616205-83/security-firm-rsa-took-millions-from-nsa-report/