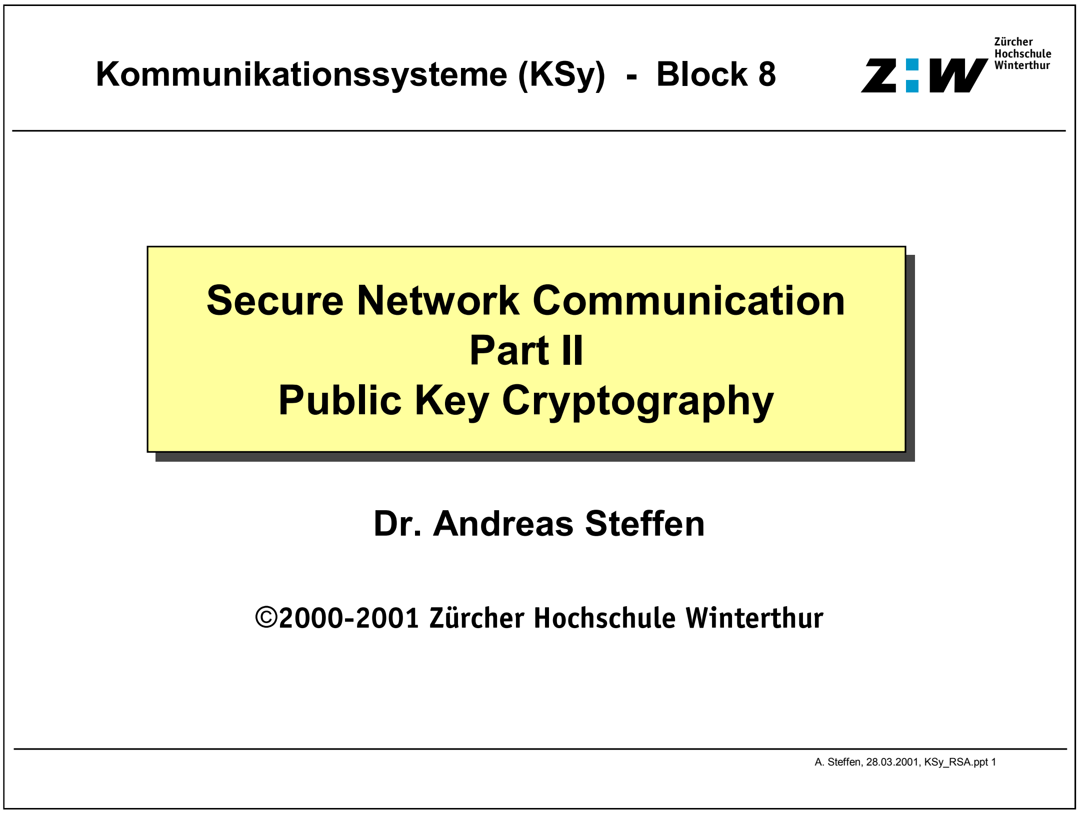


Secure Network Communication
Part II
Public Key Cryptography

Dr. Andreas Steffen

Secure Key Distribution Problem

- Bad scalability of symmetric key cryptosystems
- Public key distribution system as a solution

Public Key Cryptography

- Inventors
- Basic Principles
- One-way functions with trap doors
- Hard problems

Mathematical Operations in Finite Fields

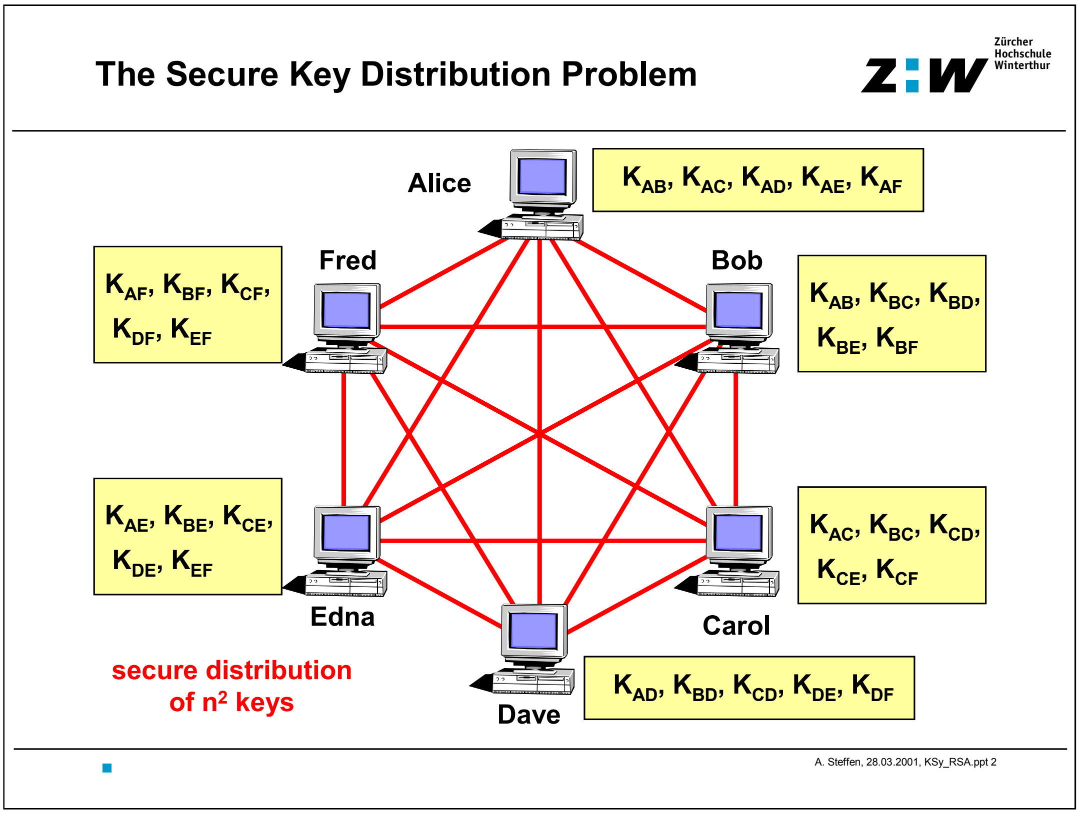
- Addition
- Negative element
- Multiplication
- Inverse element

RSA Public Key Cryptosystem

- Hard problem of factoring large numbers
- Key generation algorithm
- Public and private keys
- Encryption and decryption
- Efficient Exponentiation of large numbers
- Contest
- How to find large prime numbers

Diffie-Hellman Key-Exchange Algorithm

- Generating a common secret key



The Secure Key Distribution Problem

Zürcher Hochschule Winterthur

Alice: $K_{AB}, K_{AC}, K_{AD}, K_{AE}, K_{AF}$

Bob: $K_{AB}, K_{BC}, K_{BD}, K_{BE}, K_{BF}$

Carol: $K_{AC}, K_{BC}, K_{CD}, K_{CE}, K_{CF}$

Dave: $K_{AD}, K_{BD}, K_{CD}, K_{DE}, K_{DF}$

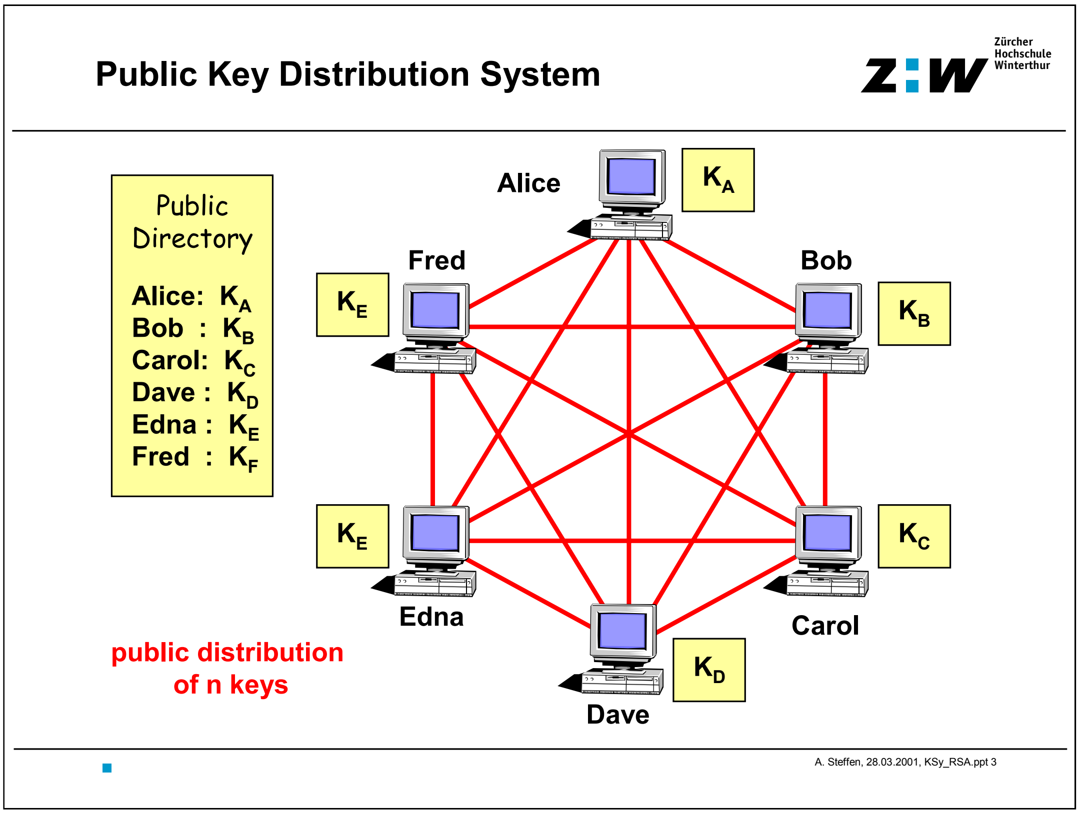
Edna: $K_{AE}, K_{BE}, K_{CE}, K_{DE}, K_{EF}$

Fred: $K_{AF}, K_{BF}, K_{CF}, K_{DF}, K_{EF}$

secure distribution of n^2 keys

Key Distribution Problem in Dense Networks

- In densely-meshed networks where many parties communicate with each other, the required number of secret keys necessary when using symmetric encryption algorithms increases quadratically with the number of participants since in a fully-meshed network to each of the *n* communication partners (*n-1*) keys must be securely delivered.
- Take as an example a broadband communications network with 100 fully-meshed nodes were each session key is changed every hour, resulting in a requirement to safely distribute about 240‘000 keys each day.
- As can easily be seen, secret key distribution scales very badly with an increasing number of participants. Therefore for a long time people had been looking for alternative ways of establishing secure connections. A very efficient solution was finally found in 1976 with the novel concept of a **Public Key Cryptosystem**.



Public Key Distribution System

Zürcher
Hochschule
Winterthur

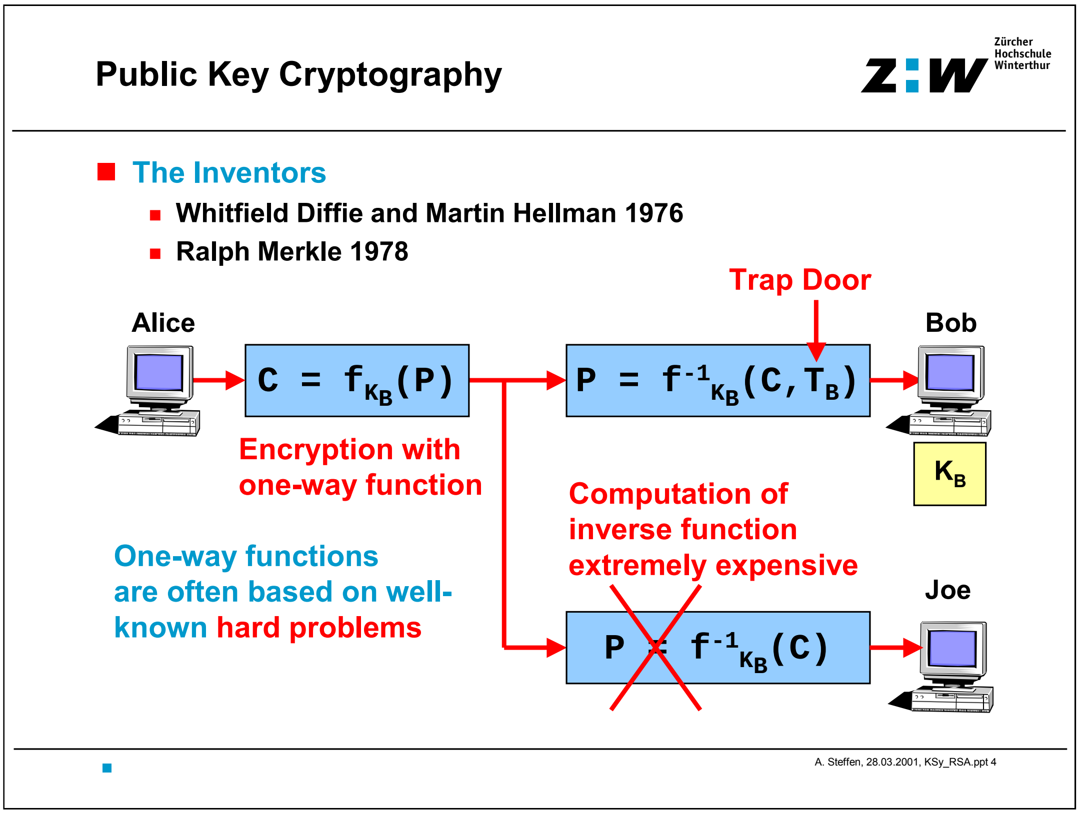
Public Directory

Alice: K_A
Bob : K_B
Carol: K_C
Dave : K_D
Edna : K_E
Fred : K_F

public distribution of n keys

Public Key Distribution System

- In a **Public Key Cryptosystem** each user or host possesses a single key pair consisting of a private key which is kept secret by the no and a matching public key which is published in a public directory (usually an LDAP or WWW server).
- If a user Alice wants to send an encrypted message to user Bob then Alice encrypts her message with Bob's public key KB fetched from the public directory and sends it to Bob. Since Bob is the only one in possession of the matching private key, he alone can decrypt the encrypted message sent to him.
- Since only the public key of the recipient is required, with *n* users only *n* distinct keys are required. Under the assumption that each user generates her own public/ private key pair locally, no secure channels are required for the distribution of the public keys, since the don't contain any secret and must be put into the public domain anyway.



Public Key Cryptography

- **The Inventors**
  - **Whitfield Diffie and Martin Hellman 1976**
  - **Ralph Merkle 1978**

Alice

$C = f_{K_B}(P)$

Encryption with one-way function

Trap Door

$P = f^{-1}_{K_B}(C, T_B)$

Bob

K_B

Computation of inverse function extremely expensive

Joe

$P \cancel{=} f^{-1}_{K_B}(C)$

One-way functions are often based on well-known hard problems

Inventors of Public Key Cryptography

- The concept of a Public Key Cryptosystem was invented at around the same time by **Whitfield Diffie**, **Martin Hellman** and **Ralph Merkle**. Whereas the first two researchers published their invention in 1976 and got all the fame, Ralph Merkle had the misfortune that the printing of his paper got delayed by more than a year so that it got published not until 1978. Today it is generally recognized that all three scientists are the fathers of public key cryptography.
- Recently it became known that already in 1970, **James Ellis**, at the time working for the British government as a member of the Communications-Electronics Security Group (CESG), formulated the idea of a Public Key Cryptosystem. Several practical algorithms including one variant very similar to RSA and another one identical to the Diffie-Hellman key exchange were discovered within the CESG. Unfortunately the British researchers were not allowed to publish their results due to state security reasons.

Basic Principles of Public Key Cryptography

- All public key cryptosystems are based on the notion of a **one-way function,** which, depending on the public key, converts plaintext into ciphertext using a relatively small amount of computing power but whose **inverse function** is extremely expensive to compute, so that an attacker is not able to derive the original plaintext from the transmitted ciphertext within a reasonable time frame.
- Another notion used in public key cryptosystems is that of a **trap door** which each one-way function possesses and which can only be activated by the legitimate user holding the private key. Using the trapdoor, decryption of the ciphertext becomes easy.
- Many public key cryptosystems are based on known **hard problems** like the factoring of large numbers into their prime factors (RSA) or taking discrete logarithms over a finite field (Diffie-Hellman).

RSA Public Key Cryptosystem

- The Inventors
  - R - Ron Rivest
  - S - Adi Shamir
  - A - Leonard Adleman
- The One-Way Function
  - The exponentiation function $y = f(x) = x^e \bmod n$ can be computed with reasonable effort.
  - Its inverse $x = f^{-1}(y)$ is extremely difficult to compute.
- The Hard Problem Securing the Trap Door
  - The RSA public key algorithm is based on the well-known hard problem of factoring large numbers into its prime factors that has been studied over many centuries.

The RSA-155 Challenge

The Effort

- 512 bit number (155 decimal digits)
- factored on August 22, 1999 after 7 months of cracking
- 300 workstations and Pentium PCs, 1 Cray supercomputer

```
10941738641570527421809707322040357612O
03732945449205990913842131476349984288 9
```

Mathematical Operations in Finite Fields

Mathematical Operations in Finite Fields

Addition

$$c = (a + b) \bmod n = (a \bmod n + b \bmod n) \bmod n$$

Example with modulus $n = 5$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

a

b

c

(7 + 9) mod 5 = 16 mod 5 = 1

(7 + 9) mod 5 = (7 mod 5 + 9 mod 5) mod 5
= (2 + 4) mod 5
= 6 mod 5 = 1

Mathematical Operations in Finite Fields

Negative Element

$y = -x \bmod n$

or

$y + x \bmod n = 0$

Example with modulus $n = 5$

x	0	1	2	3	4
y	0	4	3	2	1

Mathematical Operations in Finite Fields

Multiplication

$$c = (a \cdot b) \bmod n = (a \bmod n \cdot b \bmod n) \bmod n$$

Example with modulus $n = 5$

·	0	1	2	3	4	*a*
0	0	0	0	0	0	
1	0	1	2	3	4	
2	0	2	4	1	3	
3	0	3	1	4	2	
4	0	4	3	2	1	
b						*c*

$$(7 \cdot 9) \bmod 5 = 63 \bmod 5 = 3$$

$$\begin{aligned}(7 \cdot 9) \bmod 5 &= (7 \bmod 5 \cdot 9 \bmod 5) \bmod 5 \\ &= (\quad 2 \quad \cdot \quad 4 \quad) \bmod 5 \\ &= 8 \bmod 5 = 3\end{aligned}$$

Mathematical Operations in Finite Fields

Inverse Element

$y = x^{-1} \bmod n$ or $y \cdot x \bmod n = 1$

Example with modulus $n = 5$

x	0	1	2	3	4
y	-	1	3	2	4

RSA Public Key Cryptosystem

RSA Public Key Cryptosystem

Key Generation Algorithm

- Step 1: Choose two random large prime numbers p and q
  - For maximum security, choose p and q of about equal length, e.g. 512-1024 bits each.
- Step 2: Compute the product $n = p \cdot q$
- Step 3: Choose a random integer $e < (p-1)(q-1)$
  - The numbers e and $(p-1)(q-1)$ must be relatively prime, i.e. they should not share common prime factors.
- Step 4: Compute the unique inverse $d = e^{-1} \bmod (p-1)(q-1)$
  - The equation $d \cdot e \bmod (p-1)(q-1) = 1$ can be solved using the Euclidian algorithm.

RSA Public Key Cryptosystem
Key Generation Example

- p = 3, q = 11: **n = p·q = 33**
- (p-1)·(q-1) = 2 · 10 = 2 · 2 · 5 = 20
- the public exponent e must be relatively prime to (p-1)·(q-1) , i.e. it cannot contain any factors of 2 and 5

e	d	e·d	e·d mod 20
3	7	21	1
7	3	21	1
9	9	81	1
11	11	121	1
13	17	221	1
17	13	221	1
19	19	361	1

all possible choices for the exponents e and d

RSA Public Key Cryptosystem

Public and Private Keys

- Public Key: „modulus“ n and „public exponent“ e
  - Publish n and e in a public directory, so that anybody wanting to send you a confidential message can retrieve it.

- Private Key: „modulus“ n and „private exponent“ d
  - The private exponent d is your secret. It should be protected either by storing it in a tamper-proof smart card or when stored on a disk by encrypting it with a symmetric cipher secured by a secret passphrase of your choice.
  - The large primes p and q that were used for key generation are not needed any more and should be erased.

RSA Public Key Cryptosystem
Encryption and Decryption

- **Encryption of a plaintext block x:**
  - The sender uses the public key of the recipient to encrypt $x < n$.

$$y = x^e \bmod n$$

- **Decryption of a ciphertext block y:**
  - The recipient uses her private key to recover the plaintext block x

$$x = y^d \bmod n$$

- **Without proof:**

$$y^d = (x^e)^d = x^{e\cdot d} = x^{m\cdot(p-1)\cdot(q-1) + 1} = x^1 = x \pmod{n}$$

- **Encryption and Decryption are symmetric operations**
  - The order of the exponentiation with the public exponent e and the private exponent d can be exchanged.

RSA Public Key Cryptosystem
Encryption / Decryption Example

- **Encryption with Public Key**

$n = 33, e = 3$

Binary Plaintext	0101001001001011010011 ...			
Groups of 5 Bits	01010	01001	00101	10100 ...
Decimal Plaintext	10	9	5	20
$y = x^3$	1000	729	125	8000
$y = x^3 \bmod 33$	10	3	26	14

- **Decryption with Private Key**

$n = 33, d = 7$

Decimal Ciphertext	10	3	26	14
$x = y^7$	10^7	2187	26^7	14^7
$x = y^7 \bmod 33$	10	9	5	20

RSA Public Key Cryptosystem

Efficient Exponentiation of Large Numbers

- **Multiplication in finite fields**
  - (a·b) mod n = [(a mod n) · (b mod n)] mod n
- **Straight exponentiation method with e-1 multiplications**
  - $y = x^e = x \cdot x \cdot \ldots \cdot x \mod n$
- **Efficient exponentiation with < $2 \cdot \log_2 e$ multiplications**
  - based on the binary representation of the exponent

$$e = b_k\, 2^k + b_{k-1}\, 2^{k-1} + \ldots + b_i\, 2^i + \ldots + b_2\, 2^2 + b_1\, 2 + b_1$$

with $b_i = \{0,1\}$ and $k = \log_2 e$

$$y = \left(x^{2^k}\right)^{b_k} \cdot \left(x^{2^{k-1}}\right)^{b_{k-1}} \cdot \quad \ldots \quad \cdot \left(x^{2^2}\right)^{b_2} \cdot \left(x^2\right)^{b_1} \cdot \left(x\right)^{b_0} \mod n$$

with the iterative squaring:

$$x^{2^i} = \left(x^{2^{i-1}}\right)^2 \mod n$$

RSA Public Key Cryptosystem

Exponentiation Example

- **Encryption with Public Key**

n = 33, e = 13

  - $e = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
  - $e = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$
  - $y = x^e = (x^8)^1 \cdot (x^4)^1 \cdot (x^2)^0 \cdot (x)^1 \mod n$
  - $y = x^8 \cdot x^4 \cdot x \mod n$
  - $x^2 = x \cdot x \mod n, \; x^4 = x^2 \cdot x^2 \mod n, \; x^8 = x^4 \cdot x^4 \mod n$

- **Decryption with Private Key**

n = 33, d = 17

  - $d = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
  - $d = 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$
  - $x = y^d = (y^{16})^1 \cdot (y^8)^0 \cdot (y^4)^0 \cdot (y^2)^0 \cdot (y)^1 \mod n$
  - $x = y^{16} \cdot y \mod n$
  - $y^2 = y \cdot y \mod n, \; y^4 = y^2 \cdot y^2 \mod n, \; y^8 = y^4 \cdot y^4 \mod n,$
    $y^{16} = y^8 \cdot y^8 \mod n$

RSA Public Key Cryptosystem
Contest

- **Choose a plaintext number 1 < *x* < 33 and keep it secret!**
- **Encrypt *x* with RSA using the public key *n* = 33, *e* = 13.**
- **Exchange the encrypted number y with your neighbour.**
- **Decrypt your neighbour‘s number using the private key *n* = 33, *d* = 17.**
- **Check with your collegue if the decrypted number equals the original plaintext number.**

RSA Public Key Cryptosystem

Plaintext to Ciphertext Mapping

- n = 33, e = 13, d = 17 $y = x^e \bmod n$

x	y	x	y	x	y	x	y	x	y
0	0	8	17	16	4	24	30	32	32
1	1	9	3	17	29	25	16		
2	8	10	10	18	24	26	20		
3	27	11	11	19	28	27	15		
4	31	12	12	20	14	28	7		
5	26	13	19	21	21	29	2		
6	18	14	5	22	22	30	6		
7	13	15	9	23	23	31	25		

RSA Public Key Cryptosystem
How to find large random prime numbers

- There are 10^{151} primes 512 bits in length or less.
- There are only 10^{77} atoms in the universe.
- The chance that two people choose the same prime factors for key generation is therefore near to nil !
- To prove that a randomly chosen number is really prime you would have to factor it. Try small factors (3, 5, 7, 11, ...)
- Probabilistic Primality Tests (e.g. Rabin-Miller)

Result of Primality Test	not prime	is prime	random number is a
	100 %	0.1 %	composite number
	0 %	99.9 %	prime number

- After passing 5 tests, assume a random number to be prime

Diffie-Hellman
Key-Exchange Algorithm

The Diffie-Hellman Key-Exchange Algorithm

Generating a Common Secret Key

Zürcher Hochschule Winterthur

- Alice and Bob agree on a large prime modulus n, a primitive element g and the one-way function $y = f(x) = g^x \bmod n$.
- The integers n and g are not secret and can be published.
- Alice chooses a large random integer a and sends Bob
  $A = g^a \bmod n$
- Bob chooses a large random integer b and sends Alice
  $B = g^b \bmod n$
- Alice computes
  $s = B^a \bmod n \;=\; g^{ba} \bmod n$
- Bob computes
  $s = A^b \bmod n \;=\; g^{ab} \bmod n$
- Alice and Bob share now the secret key $s = g^{ab} \bmod n$
- Since computing the inverse $x = f^{-1}(y)$ is extremely difficult, no one listening to the key-exchange can compute the secret key s from the values A, B, n and g.

The Diffie-Hellman Key-Exchange Algorithm

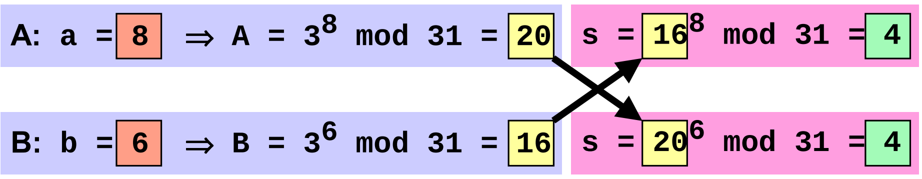
Practical Example

$$y = g^x \bmod n$$

Prime modulus: n = 31
Primitive element: g = 3

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y	1	3	9	27	19	26	16	17	20	29	25	13	8	24	10	30

x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
y	28	22	4	12	5	15	14	11	2	6	18	23	7	21	1



A: a = 8 ⇒ A = 3^8 mod 31 = 20 s = 16^8 mod 31 = 4

B: b = 6 ⇒ B = 3^6 mod 31 = 16 s = 20^6 mod 31 = 4