Public Key Cryptography

Sushmita Ruj

Recap

- Birthday paradox
- Hash Function Construction: Merkle Damgard, SHA
- Hash Function Construction: Sponge construction, SHA3
- HMAC
- Number Theory

This Lecture

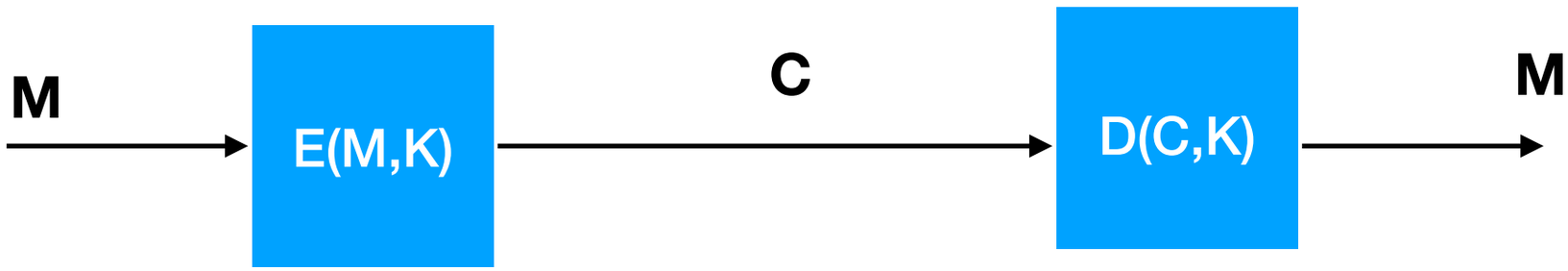
- Public Key Cryptography
- Public Key infrastructure
- Number Theory
- RSA

Symmetric Key Cryptography

K: Key

K

K: Key



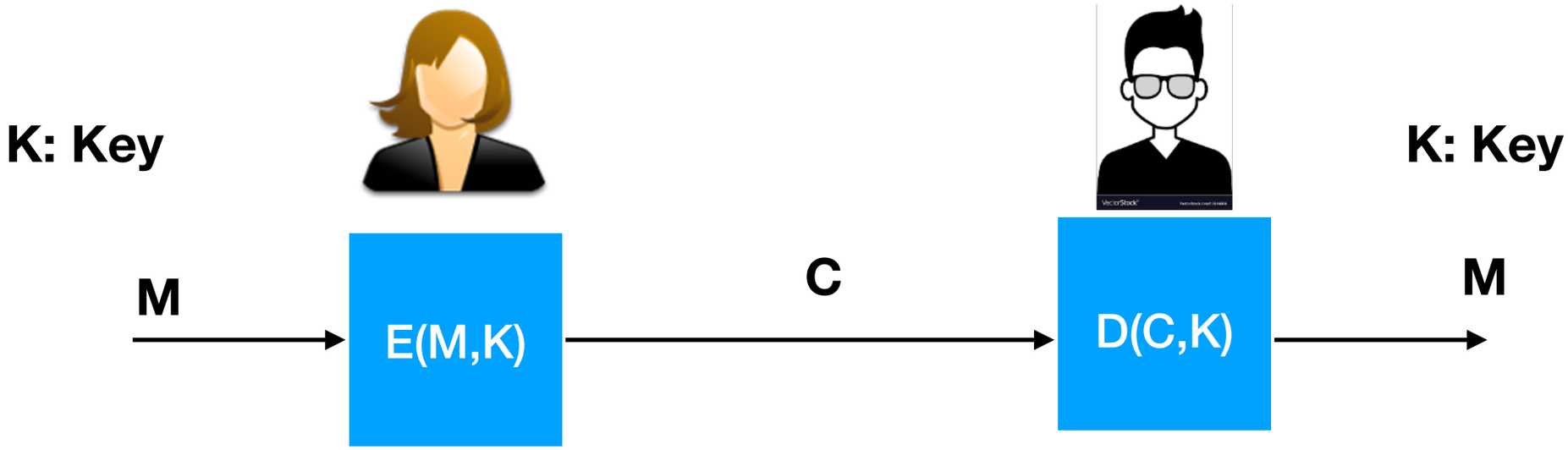
There Algorithms:

KeyGen: Key generation algorithm, generated the common key k

E(M,k): Encryption function, takes message M and Key k, generates Ciphertext C

D(C,k): Encryption function, takes cipher text C and Key k, generates Ciphertext M

Symmetric Key Cryptography

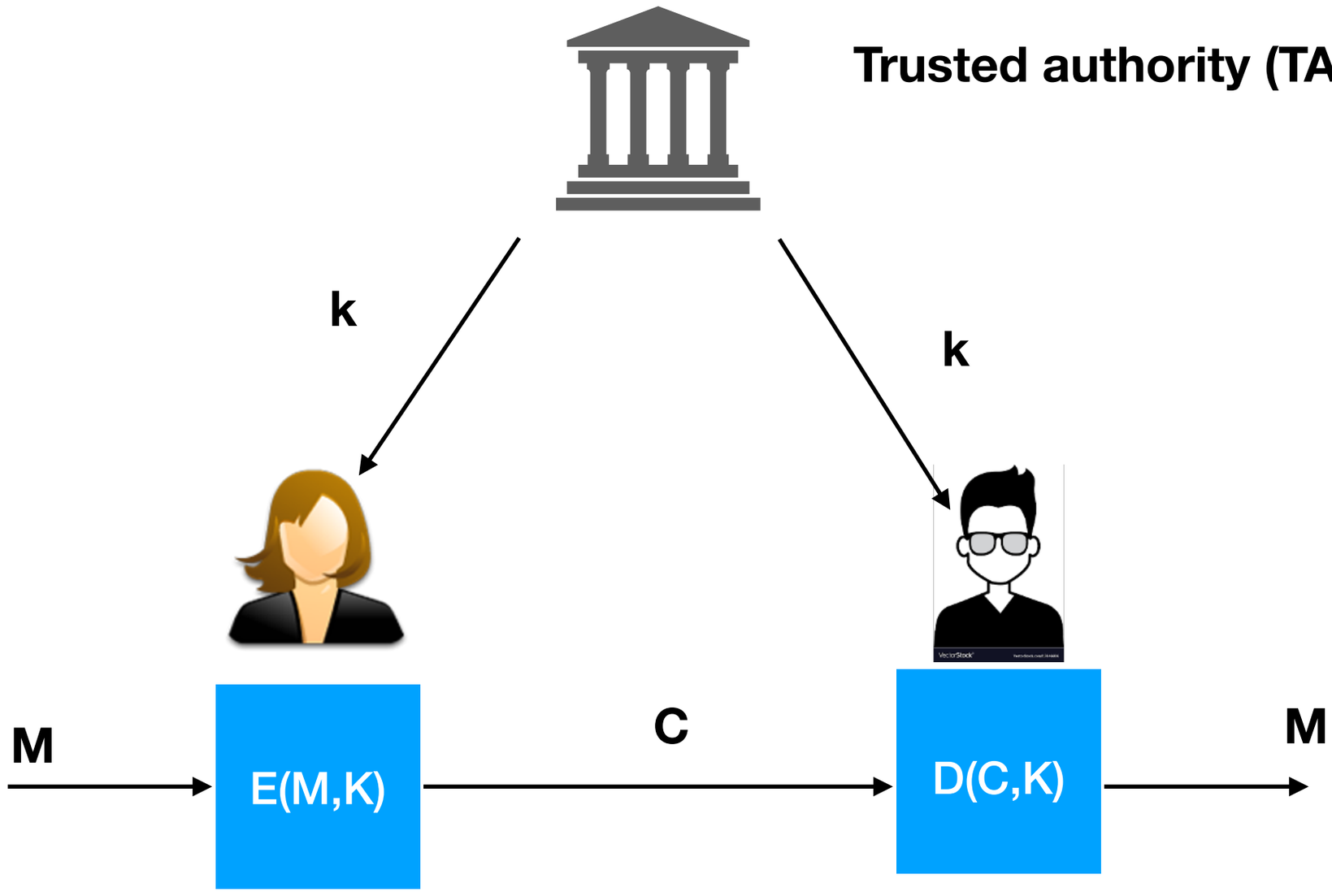


- $\varepsilon = (\mathscr{M}, \mathscr{C}, \mathscr{K})$
- $KeyGen(1^k) \rightarrow k \in \mathscr{K}$
- $For\ m \in \mathscr{M}, k \in \mathscr{K}, E(m,k) \rightarrow c$
- $D(c,k) \rightarrow m'$
- Correctness: $\forall k \in \mathscr{K}$ and messages $m \in \mathscr{M}$, if we execute $c \overset{R}{\leftarrow} E(m,k)$, $m' \overset{R}{\leftarrow} D(c,k)$, then with probability 1, $m = m'$

Challenge is to decide the common key k

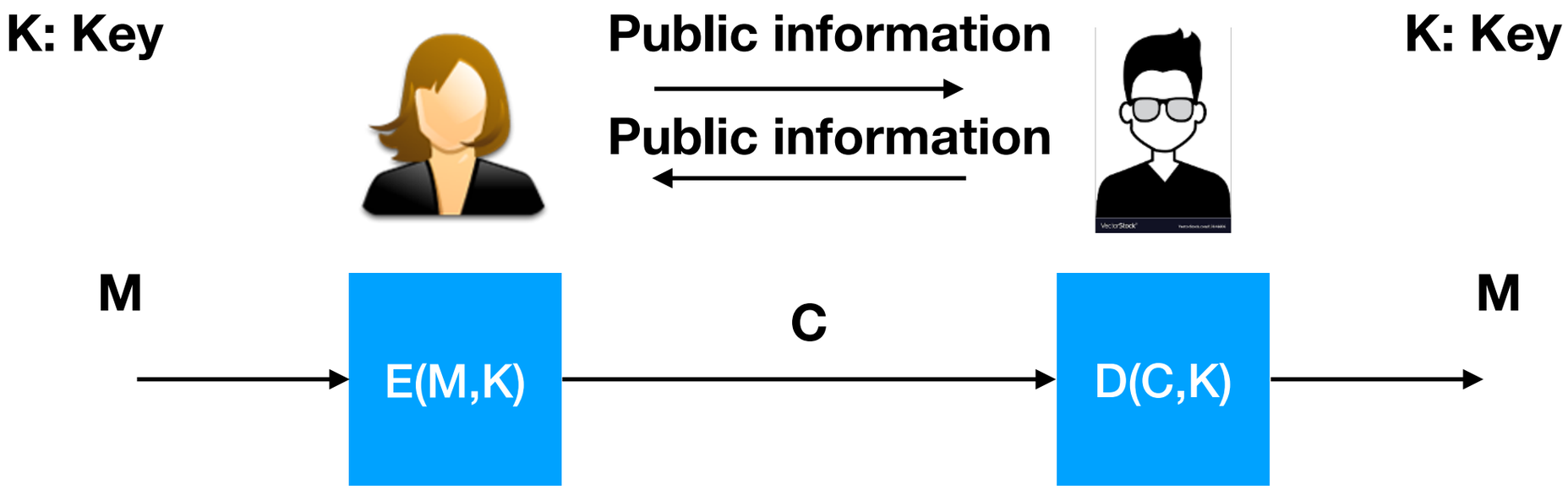
How to establish a common key

Trusted Authority Distributes keys



Problem is that TA can get corrupted, what happens then?

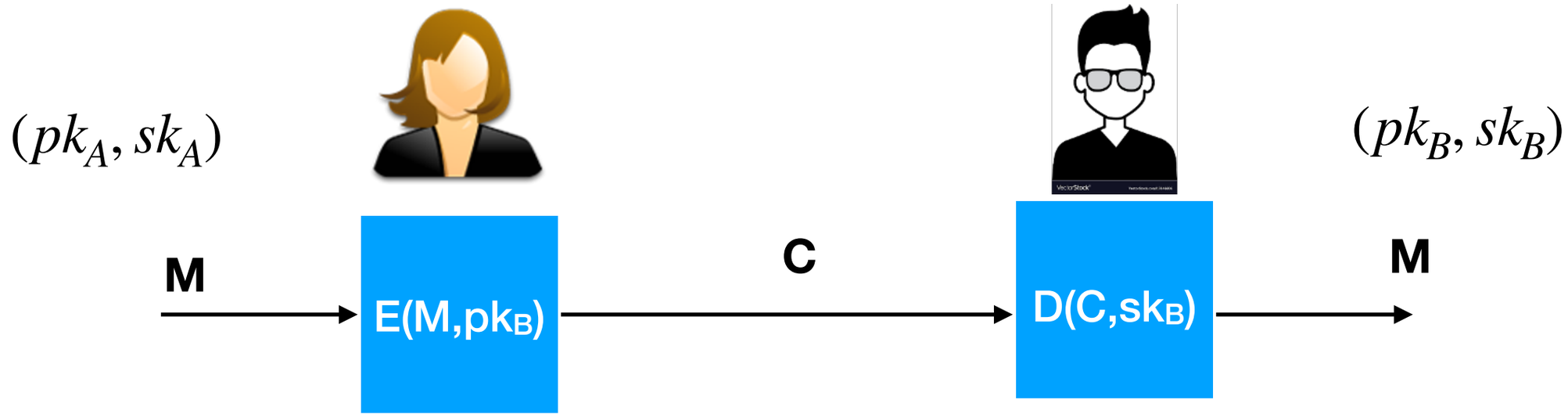
Key Agreement



- Establish shared key between Alice and Bob
- **Without** assuming an existing shared (‘master’) key or trusted authority
- Use public information from A, B to setup shared secret key k.
- Eavesdropper cannot learn the key k.

Whitfield Diffie & Martin Hellman 1976 “New Directions of Cryptography
Awarded Turing Award in 2015

Public Key Cryptography



- $\varepsilon = (\mathscr{M}, \mathscr{C}, \mathscr{K})$

Alice wants to send a message to Bob secretly

- $KeyGen(1^k) \rightarrow (pk_A, sk_A), (pk_B, sk_B), \ldots$
- $For\ m \in \mathscr{M}, E(m, pk_B) \rightarrow c$

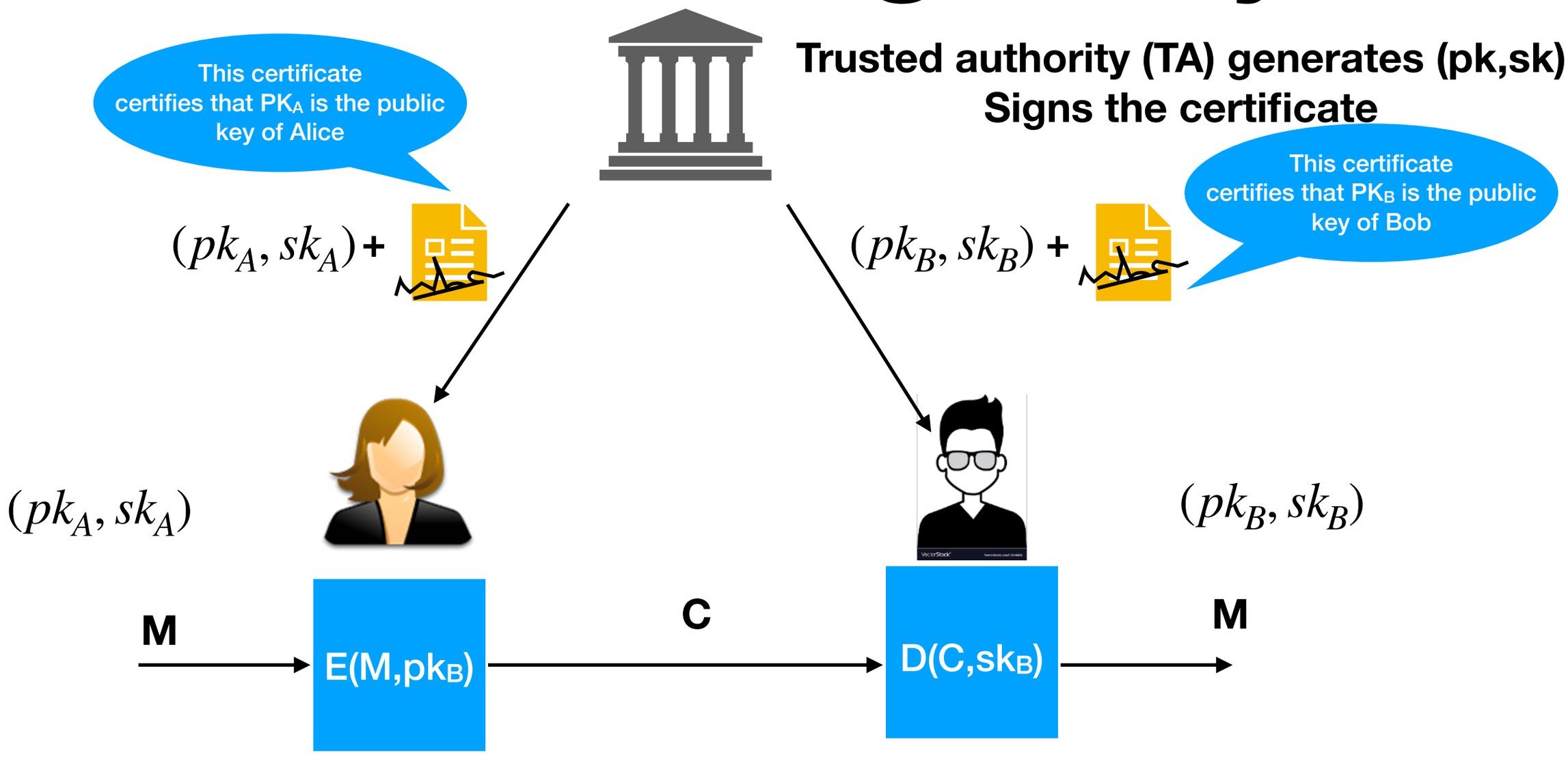
Public key of Bob known to everyone

- $D(c, sk_B) \rightarrow m'$

Secret key known only to Bob, else only Bob can decrypt

- Correctness: $\forall k \in \mathscr{K}$ and messages $m \in \mathscr{M}$, if we execute $c \xleftarrow{R} E(m, pk_B)$, $m' \xleftarrow{R} D(c, sk_B)$, then with probability 1, $m = m'$

How to Manage Keys?



TA in real are Digicert, Lets Encrypt, Identrust, GoDaddy etc.

Problem is that TA can get corrupted, what happens then?

Digital Certificate (X.509 v3)

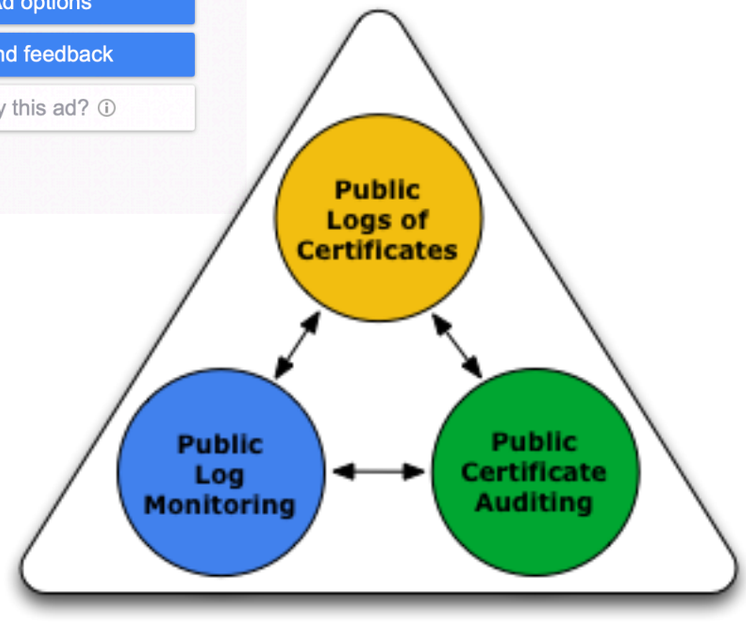
- Certificate
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before
    - Not After
  - Subject name
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

Rouge TA and Certificate Transparency

Hacking

DigiNotar SSL certificate hack amounts to cyberwar, says expert

Dutch government revokes certificates used for all its secure online transactions, while CIA, Google, Microsoft and others affected by hack called 'worse than Stuxnet'



Public Key Cryptography

- Encryption: RSA, El Gamal, EC encryption
- Authentication via signatures: RSA, DSA etc
- Advantages
  - To distribute:
    - From directory (ensure or trust authentication)
    - From incoming message (if authenticated)
    - Less keys to distribute (same public key to all)
  - To maintain:
    - Can keep in non-secure storage
    - Validate (e.g. using MAC) before using
    - Less keys: $O(|parties|)$, not $O(|parties|^2)$ *(Everypair of users have a common key in SKE)*

Why not always use Public Key Cryptography

- Assumptions
  - Applied PKC algorithms are based on a small number of specific computational assumptions
    - Mainly: hardness of factoring and discrete-log
    - Both may fail against quantum computers
- Overhead
  - Computational
  - Key length
  - Output length (ciphertext/signature)

Performance And Security of PKC

- Requires related public, private keys
  - Public key does not expose private key
- Substantial overhead
  - Successful cryptanalytic shortcuts → need long keys (cf. shared key!)
  - Elliptic Curves (EC) may allow shorter key (almost no shortcuts found)
  - Complex computations
  - RSA: very complex (slow) key generation
- Most: based on hard problems (Computationally hard problem in the average case)

[LV02]	Required key size		
Year	AES	RSA, DH	EC
2010	78	1369	160
2020	86	1881	161
2030	93	2493	176
2040	101	3214	191

Commercial-grade security
Lenstra & Verheul [LV02]

Key Sizes

Year	Symmetric			Factoring (**RSA**), DiscLog (DH)			EC		
	LV '02	NIST 2014	BSI '17	LV 2002	NIST 2014	BSI '17	LV '02	NIST 2014	BSI '17
2020	**86**	**112**	**128**	1881	2048	2000	161	224	250
2030	**93**	**112**	**128**	2493	2048	3000	176	224	250
2040	**101**	**128**	**128**	3214	3072	3000	191	256	250
Cr++	**4525 MiB/s AES/CTR 128b**			0.01ms(1024b), 0.03ms(2048b)			**1ms (256b ECIES)**		

How to use PKE

- Minimize the use of PKC
- In particular: apply PKC only to **short inputs**
- How ??

- For public-key encryption:
  - **Hybrid encryption**
    - Choose Secret key k and send use PKE to encrypt secret key
    - Then use SKE with key k to encrypt the message

Basic Number Theory

Notations

Def: $\underline{Z^*_N}$ = (set of invertible elements in Z_N) =

$= \{ x \in Z^*_N : \gcd(x,N) = 1 \}$

Examples:

1. for prime p, $Z^*_p = Z_p - \{0\} = \{1, 2, \ldots p-1\}$
2. $Z^*_{12} = \{ 1, 5, 7, 11\}$

For x in Z^*_N, can find x^{-1} using extended Euclid algorithm.

Modular Inversion

Over the rationals, inverse of 2 is ½ . What about in Z_N ?

Def: The **inverse** of x in is an element y in s.t. x.y =1
y is denoted x^{-1} .
Examples: 7^{-1} mod 11 = ?

Example: let N be an odd integer. The inverse of 2 in is

Lemma: x in Z_N has an inverse if and only if gcd(x,N) = 1
Proof:
gcd(x,N)=1 ⇒ ∃ a,b: $a \cdot x + b \cdot N = 1$, => a.x =1 => $x^{-1} = a$

If gcd(x,N) >1, a.x not= 1, so no inverse.

then say, gcd (x, N) = 2, so, for all a, gcd(a,N) is even.

GCD

Def: For ints. x,y: **gcd(x, y)** is the greatest common divisor of x,y

Example: gcd(24, 18) = 6

Fact: for all ints. x,y there exist ints. a,b such that

$$\mathbf{a \cdot x + b \cdot y = gcd(x,y)}$$

a,b can be found efficiently using the extended Euclid alg.

If gcd(x,y)=1 we say that x and y are **relatively prime**

Euclidean Algorithms

Algorithm 6.1: EUCLIDEAN ALGORITHM(a, b)

$r_0 \leftarrow a$
$r_1 \leftarrow b$
$m \leftarrow 1$
while $r_m \neq 0$
do $\begin{cases} q_m \leftarrow \lfloor \frac{r_{m-1}}{r_m} \rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m + 1 \end{cases}$
$m \leftarrow m - 1$
return $(q_1, \ldots, q_m; r_m)$
comment: $r_m = \gcd(a, b)$

Extended Euclidean Algorithm

Algorithm 6.2: EXTENDED EUCLIDEAN ALGORITHM(a, b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$
do
$temp \leftarrow t_0 - qt$
$t_0 \leftarrow t$
$t \leftarrow temp$
$temp \leftarrow s_0 - qs$
$s_0 \leftarrow s$
$s \leftarrow temp$
$a_0 \leftarrow b_0$
$b_0 \leftarrow r$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
$r \leftarrow b_0$
return (r, s, t)
comment: $r = \gcd(a, b)$ and $sa + tb = r$

Fermat's Little Theorem (1640)

Thm: Let p be a prime

$$\forall\, \mathbf{x} \in (\mathbf{Z_p})^* : \quad \mathbf{x^{p-1} = 1 \text{ in } Z_p}$$

Example: p=5. $3^4 = 81 = 1$ in Z_5

So: $x \in (Z_p)^* \Rightarrow x \cdot x^{p-2} = 1 \Rightarrow x^{-1} = x^{p-2}$ in Z_p

another way to compute inverses, but less efficient than Euclid

Application: Generating Random Primes

Suppose we want to generate a large random prime

say, prime p of length 1024 bits (i.e. $p \approx 2^{1024}$)

Step 1: choose a random integer $p \in [\ 2^{1024}\ ,\ 2^{1025}-1\ ]$

Step 2: test if $2^{p-1} = 1$ in Z_p

If so, output p and stop. If not, goto step 1 .

Simple algorithm but inefficient. **Pr[p not prime] < 2^{-60}**

Groups

- Let $G = (S, \circ)$ be a group, then the following hold
- $\forall a, b, c, (a \circ b) \circ c = a \circ (b \circ c)$ (Associativity)
- $\forall a, \exists e, s.t., a \circ e = e \circ a = a$ (e is called the identity element, identity element is unique)
- $\forall a \in S, \exists a^{-1} \in S, s.t., a \circ a^{-1} = a^{-1} \circ a = e$ (inverse exists, inverse is unique)
- Which of these are groups (Z_7, *), (Z_6, *), (Z_{12}, +)?

Cyclic Groups

Thm (Euler): $(Z_p)^*$ is a **cyclic group**, that is

$\exists\ g \in (Z_p)^*$ such that $\{1, g, g^2, g^3, \ldots, g^{p-2}\} = (Z_p)^*$

g is called a **generator** of $(Z_p)^*$

Example: p=7. $\{1, 3, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\} = (Z_7)^*$

Not every elem. is a generator: find generators of $(Z_7)^*$

Order

For $g \in (Z_p)^*$ the set $\{1, g, g^2, g^3, \ldots\}$ is called

the **group generated by g**, denoted $\langle g \rangle$

Def: the **order** of $g \in (Z_p)^*$ is the size of $\langle g \rangle$

$ord_p(g) = |\langle g \rangle| =$ (smallest $a>0$ s.t. $g^a = 1$ in Z_p)

Examples: $ord_7(3) = 6$; $ord_7(2) = 3$; $ord_7(1) = 1$

Thm (Lagrange): $\forall g \in (Z_p)^*$: **$ord_p(g)$** divides $p-1$

Euler's Phi Function

Def: For an integer N define $\phi(N) = |(Z_N)^*|$ **(Euler's ϕ func.**

Or Totient Function)

Examples: $\phi(15) =$ $|\{1,2,4,7,8,11,13,14\}| = 8.$

$\phi(p) =$ $p-1$, p is prime

For $N=p \cdot q$: $\phi(N) = N-p-q+1 = (p-1)(q-1)$

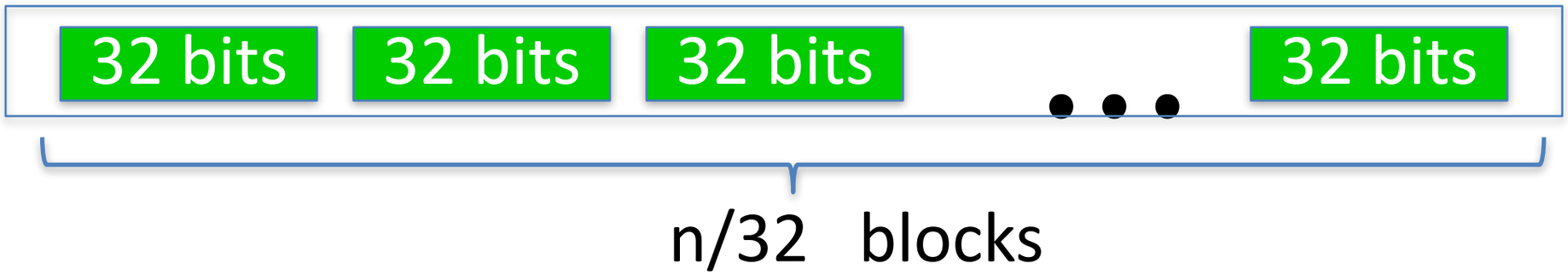
Thm (Euler): $\forall$ **x** $\in$ $\mathbf{(Z_N)^*}$ **:** $\mathbf{x^{\phi(N)} = 1}$ **in** $\mathbf{Z_N}$

Example: $5^{\phi(12)} = 5^4 = 625 = 1$ in Z_{12}

Generalization of Fermat. Basis of the RSA cryptosystem

Representing bignums

Representing an n-bit integer (e.g. n=2048) on a 64-bit machine



Note: some processors have 128-bit registers (or more) and support multiplication on them

Arithmetic

Given: two n-bit integers

- **Addition and subtraction**: $O(n)$
- **Multiplication**: $O(n^2)$
  - Karatsuba (1960): $O(n^{1.585})$
    - Basic idea: $(2^b x_2+x_1) \times (2^b y_2+ y_1)$ with 3 mults.
    - Best (asymptotic) algorithm: about $O(n \cdot \log n)$. Harvey-van der Hoeven
- **Division with remainder**: $O(n^2)$.

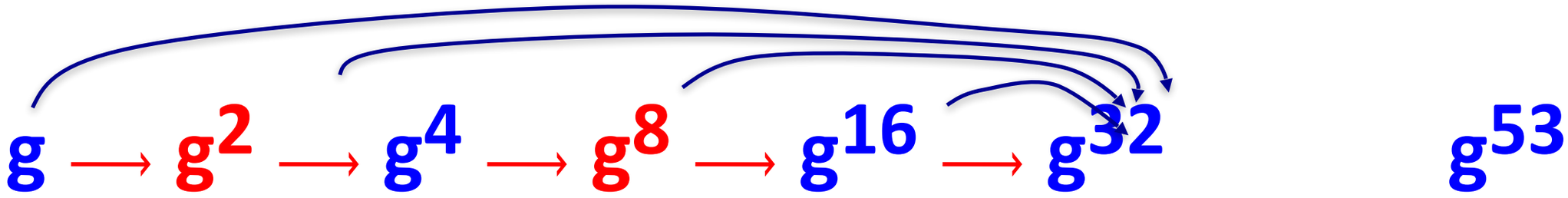
Exponentiation

Finite cyclic group G (for example $G = (Z_p)^*$)

Goal: given g in G and x compute g^x

Example: suppose $x = 53 = (110101)_2 = 32+16+4+1$

Then: $g^{53} = g^{32+16+4+1} = g^{32} \cdot g^{16} \cdot g^4 \cdot g^1$



Square and Multiple Algo

Input: g in G and x>0 ; **Output**: g^x

write $x = (x_n\, x_{n-1} \ldots x_2\, x_1\, x_0)_2$

$y \leftarrow g \quad , \quad z \leftarrow 1$

for i = 0 to n do:

if (x[i] == 1): $z \leftarrow z \cdot y$

$y \leftarrow y^2$

output z

example: g^{53}

y	z
g^2	g
g^4	g
g^8	g^5
g^{16}	g^5
g^{32}	g^{21}
g^{64}	**g^{53}**

Running time

Given n-bit int. N:

- **Addition and subtraction in Z_N**: linear time $T_+ = O(n)$

- **Modular multiplication in Z_N**: naively $T_\times = O(n^2)$

- **Modular exponentiation in Z_N (g^x)**:

$$O\big((\log x) \cdot T_\times \big) \leq O\big((\log x) \cdot n^2 \big) \leq O(n^3)$$

Tractable Problems

- Finding gcd
- Polynomial Evaluation
- Matrix multiplication
- Primality testing: Determine if a number is prime or composite (AKS Algorithm, Neeraj Kayal and Nitin Saxena were 4th Year UG students when they solved it !!!)

The factoring problem

Best known alg. (NFS): run time exp($\tilde{O}(\sqrt[3]{n})$) for n-bit integer

Current world record: **RSA-768** (232 digits)

- Work: two years on hundreds of machines
- Factoring a 1024-bit integer: about 1000 times harder

$\Rightarrow$ likely possible this decade

The RSA Algorithm

KeyGen: Choose random primes p,q ≈1024 bits. Set **N=pq**.

Choose integers **e , d s.t. e·d = 1 (mod ϕ(N))**

output pk = (N, e), sk = (N, d)

Encrypt E(x,pk): $c=x^e \mod N$

Decrypt D(x,sk,N):$c^d \mod N$

Correctness: output of decryption is $x^{ed} = x^{k\varphi(N)+1} = (x^{\varphi(N)})^k \cdot x = x$
(by Euler's thm, gcd(x, Z^*_N), $x^{\varphi(N)} = 1$ in Z_N)

HW: Try out public key encryption and key management using OpenSSL

Next Lecture

- Security of RSA: textbook RSA is not secure
- Pratical implementations of RSA
- El Gamal Encryption
- Other Hard problems and corresponding PKE

Thank you