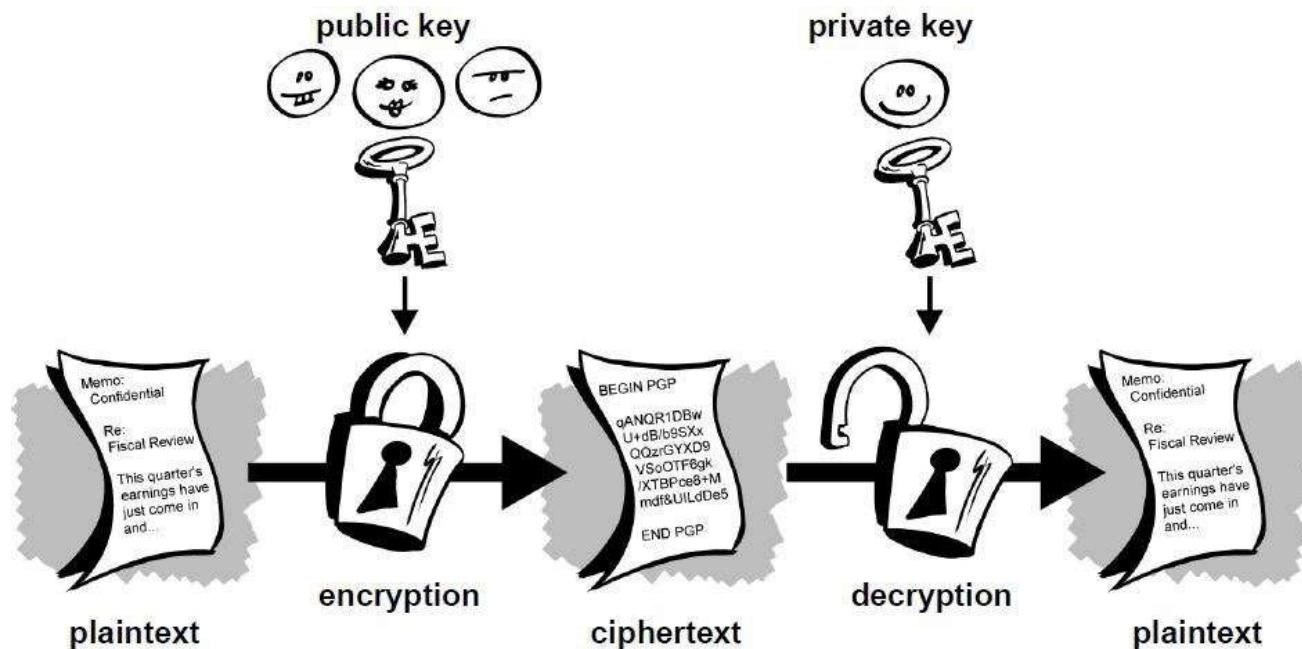


I 5-25 I

# Great Theoretical Ideas in Computer Science

## Lecture 25: Cryptography



# What is cryptography about?

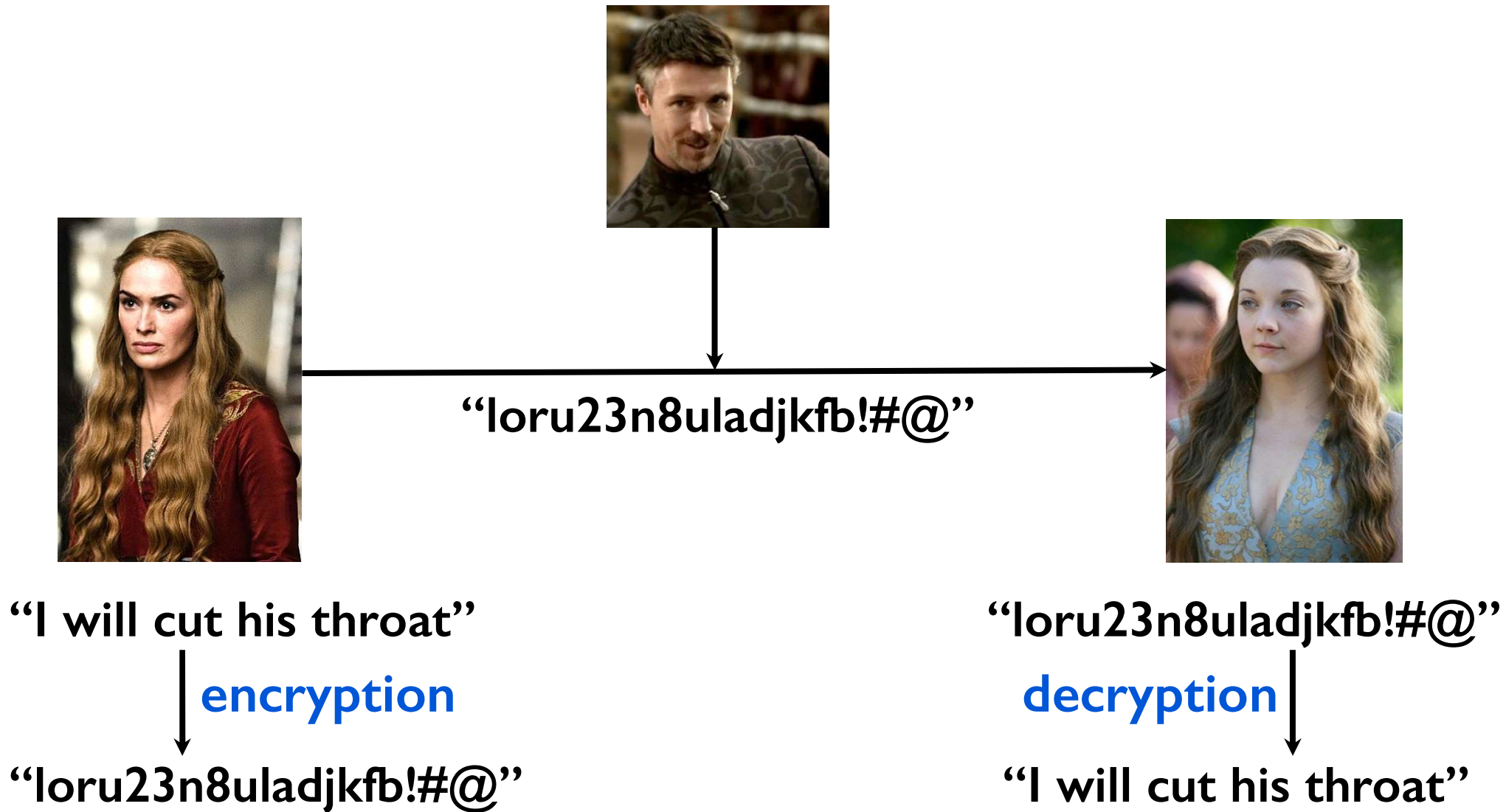
**Adversary  
Eavesdropper**



**“I will cut his throat”**

**“I will cut his throat”**

# What is cryptography about?



# What is cryptography about?

**Study of protocols that avoid the bad affects of adversaries.**

- **Secure online voting schemes?**
- **Digital signatures.**
- **Computation on encrypted data?**
- **Zero-Knowledge Interactive Proofs:**  
**Can I convince you that I have proved  $P=NP$  without giving you any information about the proof?**  
**:**

# Reasons to like cryptography

**Can do pretty cool and unexpected things.**

**Has many important applications.**

**Is fundamentally related to computational complexity.**

**In fact, comp. complexity revolutionized cryptography.**

**Applications of computationally hard problems.**

**Uses cool math (e.g. number theory).**

# The plan

First, we will review **modular arithmetic**.

Then we'll talk about **private (secret) key** crypto.

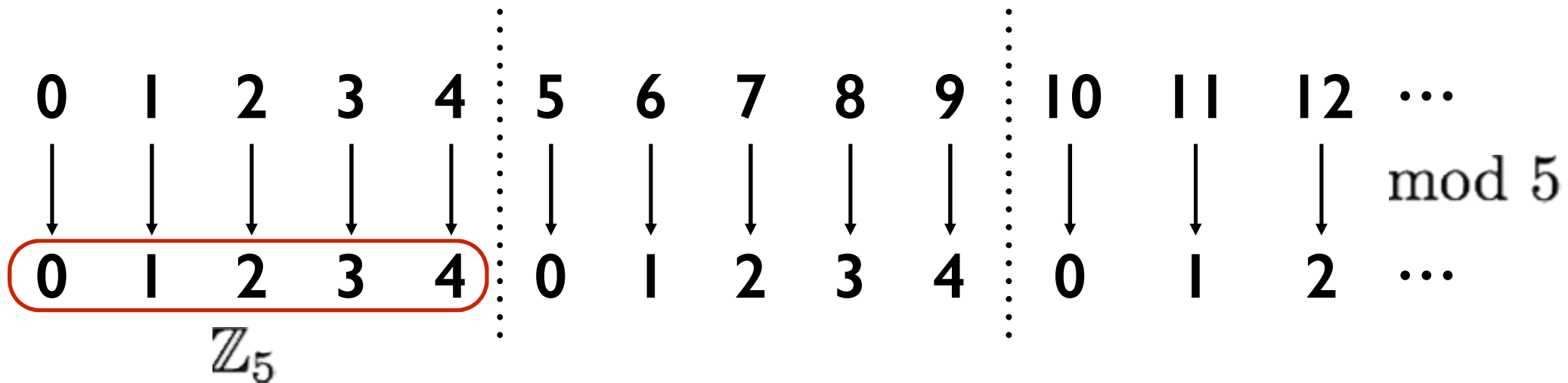
Finally, we'll talk about **public key** cryptography.

# **Review of Modular Arithmetic**

$A \bmod N$  = remainder when you divide  $A$  by  $N$

**Example**

$$N = 5$$



**We write**  $A \equiv B \bmod N$  **or**  $A \equiv_N B$

**when**  $A \bmod N = B \bmod N$ .

**Can view the universe as**  $\mathbb{Z}_N = \{0, 1, 2, \dots, N - 1\}$ .



$\mathbb{Z}_4$ 

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

$$\mathbb{Z}_N = \{0, 1, 2, \dots, N - 1\}$$

behaves nicely  
with respect to  
addition

 $\mathbb{Z}_8^*$ 

•	1	3	5	7
1	1	3	5	7
3	3	1	7	5
5	5	7	1	3
7	7	5	3	1

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

behaves nicely  
with respect to  
multiplication

$$\varphi(N) = |\mathbb{Z}_N^*|$$

$$\text{if } P \text{ prime, } \varphi(P) = P - 1$$

$$\text{if } P, Q \text{ distinct primes, } \varphi(PQ) = (P - 1)(Q - 1)$$

$$\mathbb{Z}_5^*$$

•

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

$$\varphi(5) = 4$$

$$1^0 \quad 1^1 \quad 1^2 \quad 1^3 \quad 1^4 \quad 1^5 \quad 1^6 \quad 1^7 \quad 1^8$$

$$1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$$

$$2^0 \quad 2^1 \quad 2^2 \quad 2^3 \quad 2^4 \quad 2^5 \quad 2^6 \quad 2^7 \quad 2^8$$

$$1 \quad 2 \quad 4 \quad 3 \quad 1 \quad 2 \quad 4 \quad 3 \quad 1$$

$$3^0 \quad 3^1 \quad 3^2 \quad 3^3 \quad 3^4 \quad 3^5 \quad 3^6 \quad 3^7 \quad 3^8$$

$$1 \quad 3 \quad 4 \quad 2 \quad 1 \quad 3 \quad 4 \quad 2 \quad 1$$

$$4^0 \quad 4^1 \quad 4^2 \quad 4^3 \quad 4^4 \quad 4^5 \quad 4^6 \quad 4^7 \quad 4^8$$

$$1 \quad 4 \quad 1 \quad 4 \quad 1 \quad 4 \quad 1 \quad 4 \quad 1$$

2 and 3 are called **generators**.

$$\mathbb{Z}_5^*$$

•	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

$$\varphi(5) = 4$$



$$\forall A, \quad A^4 = 1$$

$$\implies A^{4k} = (A^4)^k = 1$$

$1^0$	$1^1$	$1^2$	$1^3$	$1^4$	$1^5$	$1^6$	$1^7$	$1^8$
1	1	1	1	1	1	1	1	1
$2^0$	$2^1$	$2^2$	$2^3$	$2^4$	$2^5$	$2^6$	$2^7$	$2^8$
1	2	4	3	1	2	4	3	1
$3^0$	$3^1$	$3^2$	$3^3$	$3^4$	$3^5$	$3^6$	$3^7$	$3^8$
1	3	4	2	1	3	4	2	1
$4^0$	$4^1$	$4^2$	$4^3$	$4^4$	$4^5$	$4^6$	$4^7$	$4^8$
1	4	1	4	1	4	1	4	1

## Euler's Theorem:

For any  $A \in \mathbb{Z}_N^*$  ,  $A^{\varphi(N)} = 1$  .

## Fermat's Little Theorem:

Let  $P$  be a prime. For any  $A \in \mathbb{Z}_P^*$  ,  $A^{P-1} = 1$ .

1

||

$A^0$        $A^1$        $A^2$        $\dots$        $A^{\varphi(N)-1}$

||

||

||

||

$A^{\varphi(N)}$        $A^{\varphi(N)+1}$        $A^{\varphi(N)+2}$        $\dots$        $A^{2\varphi(N)-1}$

||

||

||

||

$A^{2\varphi(N)}$        $A^{2\varphi(N)+1}$        $A^{2\varphi(N)+2}$        $\dots$        $A^{3\varphi(N)-1}$

# IMPORTANT

---

When exponentiating elements  $A \in \mathbb{Z}_N^*$  ,  
can think of the exponent living in the universe  $\mathbb{Z}_{\varphi(N)}$  .

---

# **Algorithms for Modular Arithmetic**

> **addition**  $A + B \bmod N$

Do regular addition. Then take mod N.

> **subtraction**  $A - B = A + (-B) \bmod N$

$-B = N - B$ . Then do addition.

> **multiplication**  $A \cdot B \bmod N$

Do regular multiplication. Then take mod N.

> **division**  $A/B = A \cdot B^{-1} \bmod N$

Find  $B^{-1}$ . Then do multiplication.

> **exponentiation**  $A^B \bmod N$

> **addition**  $A + B \bmod N$

Do regular addition. Then take mod N.

> **subtraction**  $A - B = A + (-B) \bmod N$

$-B = N - B$ . Then do addition.

> **multiplication**  $A \cdot B \bmod N$

Do regular multiplication. Then take mod N.

> **division**  $A/B$

Find  $B^{-1}$ . Then

$B^{-1}$  exists iff  $\gcd(B, N) = 1$ .

Our modification of Euclid's Alg. computes  $B^{-1}$  given B and N.

> **exponentiation**



> **addition**  $A + B \bmod N$

Do regular addition. Then take mod N.

> **subtraction**  $A - B = A + (-B) \bmod N$

$-B = N - B$ . Then do addition.

> **multiplication**  $A \cdot B \bmod N$

Do regular multiplication. Then take mod N.

> **division**  $A/B = A \cdot B^{-1} \bmod N$

Find  $B^{-1}$ . Then do multiplication.

> **exponentiation**  $A^B \bmod N$

repeatedly square and mod to compute powers of two  
then multiply those mod n as necessary

> **addition**  $A + B \bmod N$

Do regular addition. Then take mod N.

> **subtraction**  $A - B = A + (-B) \bmod N$

$-B = N - B$ . Then do addition.

> **multiplication**

Do regular

> **division**

Find the

**What about roots and  
logarithms?**

> **exponentiation**  $A^B \bmod N$

repeatedly square and mod to compute powers of two  
then multiply those mod n as necessary

## Arithmetic in $\mathbb{Z}$



too big

### Two inverse functions:



???



???

## Arithmetic in $\mathbb{Z}$



too big

### Two inverse functions:



easy



easy

In  $\mathbb{Z}$



easy

$(1881676371789154860897069, 3) \longrightarrow 123456789$

(do binary search and exponentiation)



easy

$(48519278097689642681155855396759336072749841943521979872827, 3)$

$\longrightarrow 123$

(keep dividing by B)

## Arithmetic in $\mathbb{Z}_N^*$

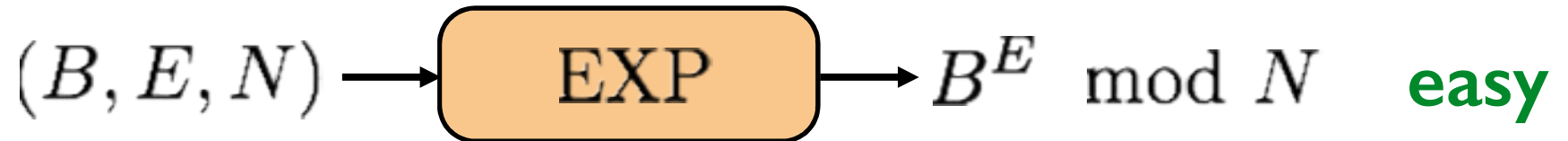
$$(B, E, N) \longrightarrow \boxed{\text{EXP}} \longrightarrow B^E \bmod N \quad \text{easy}$$

Two inverse functions:

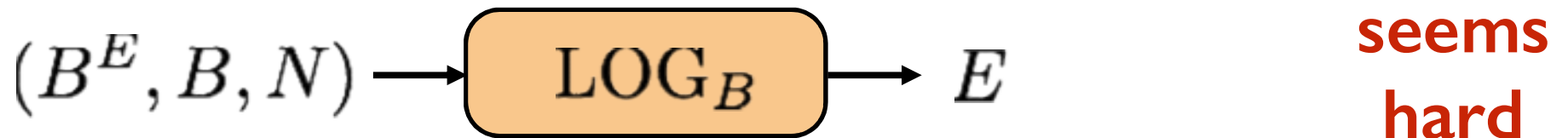
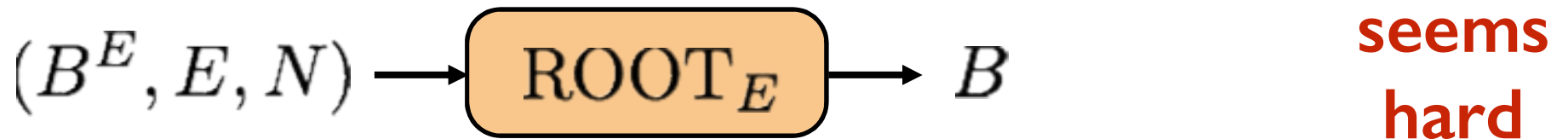
$$(B^E, E, N) \longrightarrow \boxed{\text{ROOT}_E} \longrightarrow B \quad ???$$

$$(B^E, B, N) \longrightarrow \boxed{\text{LOG}_B} \longrightarrow E \quad ???$$

Arithmetic in  $\mathbb{Z}_N^*$

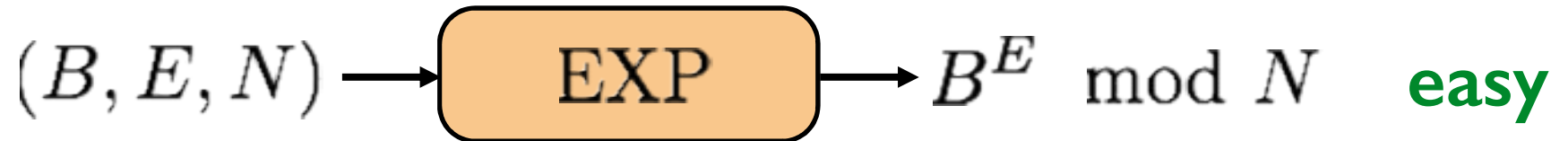


Two inverse functions:

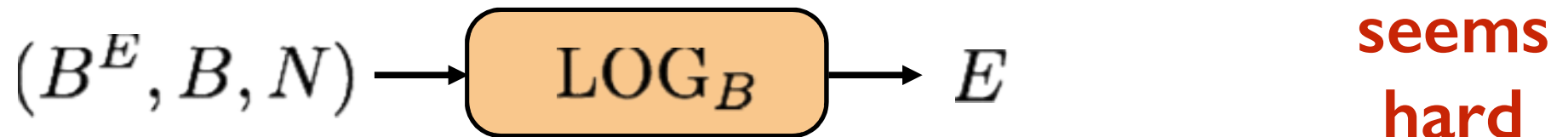
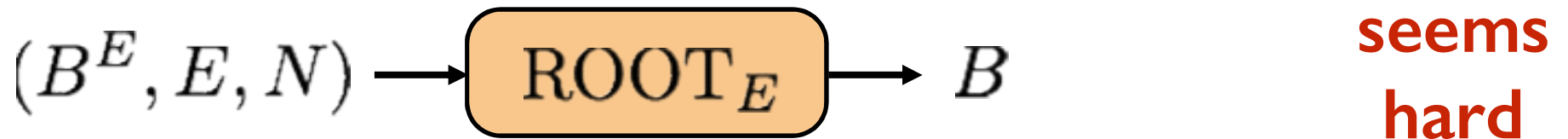


Question: Why do the algorithms from the setting of  $\mathbb{Z}$  do not work in  $\mathbb{Z}_N^*$  ?

Arithmetic in  $\mathbb{Z}_N^*$



Two inverse functions:



One-way function: easy to compute, hard to invert.  
EXP seems to be one-way.



# **Private Key Cryptography**

# Private key cryptography



**Parties must agree on a key pair beforehand.**

# Private key cryptography



**there must be a secure way of  
exchanging the key**

# Private key cryptography



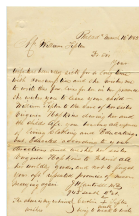
EMUPPZLRFAXYUQJZLDRSSNGNFIJ  
VQQUXGNGVYUILLTRVYQTMKTRMDP  
VFQJREZFWZTFYQWERKUTOPANCE  
GOWREKTMPTTQZGOMLAPFZRMBA  
TIMVREJADQVKGQDAGVFPFJNGRUSA  
QGEGLQGVSEESTQDLMAGCTATDWAR  
VIEZTKZMVDPKRIKFWRUWQZLZFTI  
REDDOJRTOMKQCUQWVYFTTCGEESE  
EVLJNFEHQUILLTUSWFFULLAIX  
FLQSTETKZARTDAVGGQIFUFXRDRKP  
FQWKTPLAGNIPDQGLAMVNEPFA  
ELZVIRKFTVQEXDMVYFNFAXELORE  
DNQWPKULLPABJLGLMGTVPATC  
DNUMERDNDAPMGENDPLERWILLARTQ  
DNVAPRQKREHGOPTGOMDSSNAAA  
CIBTQJTLINLILNODGOMAWALE  
TPNGATHNRAHPELNLLELPIACAE  
WNTWQHTREBARCTENSTUDETNAQOK  
EFGSEPTWENRAGIITTAHEGNCIATCR  
TEETDQSPQWVYKXAMALAKKXUJH  
REEDONIAARTTMTTEWPRDQAGRWFFB  
ACCTDMILCIBSTEGORAGSDOTDLOIT  
RLMLEAGTDHARDPNEOMGMPFEURE  
SOMMIFRAGLNSLFTYDQWVQREK  
QJQKUBJLHULIMWYLFYQWVYKXU  
TWTQSGRREZZWATJRLQDIWIFRNT  
VTTFQPRWQKZXTJQDKUTBLAUEKAR

$C$



$K_A$

$M$  (plaintext)



$K_B$

$(M, K_A)$

Enc should be “one-way”.

$(C, K_B)$

Enc

Try to ensure it using  
the secrecy of the key.

Dec

$C$  (ciphertext)

$M$

EMUPPZLRFAXYUQJZLDRSSNGNFIJ  
VQQUXGNGVYUILLTRVYQTMKTRMDP  
VFQJREZFWZTFYQWERKUTOPANCE  
GOWREKTMPTTQZGOMLAPFZRMBA  
TIMVREJADQVKGQDAGVFPFJNGRUSA  
QGEGLQGVSEESTQDLMAGCTATDWAR  
VIEZTKZMVDPKRIKFWRUWQZLZFTI  
REDDOJRTOMKQCUQWVYFTTCGEESE  
EVLJNFEHQUILLTUSWFFULLAIX  
FLQSTETKZARTDAVGGQIFUFXRDRKP  
FQWKTPLAGNIPDQGLAMVNEPFA  
ELZVIRKFTVQEXDMVYFNFAXELORE  
DNQWPKULLPABJLGLMGTVPATC  
DNUMERDNDAPMGENDPLERWILLARTQ  
DNVAPRQKREHGOPTGOMDSSNAAA  
CIBTQJTLINLILNODGOMAWALE  
TPNGATHNRAHPELNLLELPIACAE  
WNTWQHTREBARCTENSTUDETNAQOK  
EFGSEPTWENRAGIITTAHEGNCIATCR  
TEETDQSPQWVYKXAMALAKKXUJH  
REEDONIAARTTMTTEWPRDQAGRWFFB  
ACCTDMILCIBSTEGORAGSDOTDLOIT  
RLMLEAGTDHARDPNEOMGMPFEURE  
SOMMIFRAGLNSLFTYDQWVQREK  
QJQKUBJLHULIMWYLFYQWVYKXU  
TWTQSGRREZZWATJRLQDIWIFRNT  
VTTFQPRWQKZXTJQDKUTBLAUEKAR

# A note about security

**Better to consider worst-case conditions.**

**Assume the adversary knows everything except the key(s) and the message:**

**Completely sees cipher text  $C$ .**

**Completely knows the algorithms **Enc** and **Dec**.**

# Caesar shift

## Example: shift by 3

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c



(similarly for capital letters)

“Dear Math, please grow up and solve your own problems.”

↓  
“Ghdu Pdwk, sohvh jurz xs dqg vroyh brxu rzq sureohpv.”



: the shift number

Easy to break.

# Substitution cipher

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
j	k	b	d	e	l	m	c	f	g	n	o	x	y	r	s	v	w	z	a	t	u	p	q	h	i



: permutation of the alphabet

**Easy to break by looking at letter frequencies.**

# Vigenère cipher

**M** = “Dear Math, please grow up and solve your own problems.”

**K** = “helloworldhelloworldhelloworldhelloworldhell”

**K[i]** determines the shifting factor for **M[i]**.

a      shift by 0

b      shift by 1

c      shift by 2

d      shift by 3

...      ...

**A series of different Caesar  
ciphers  
based on the letters of the key.**

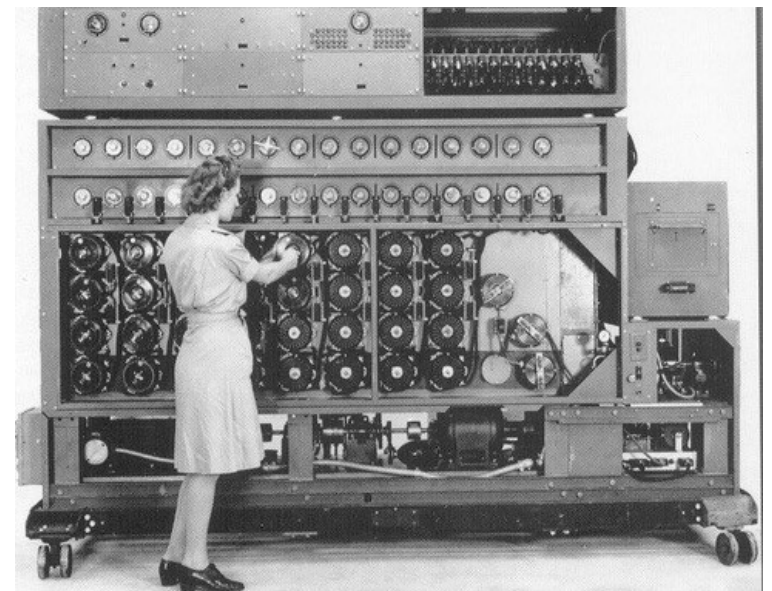
**A form of polyalphabetic cipher.**

**Easy to break.**



# Enigma

A much more complex cipher.



# One-time pad

**M** = message      **K** = key      **C** = encrypted message  
(everything in binary)

## Encryption:

$$\begin{array}{rcl} \text{M} & = & 01011010111010100000111 \\ \oplus \text{K} & = & 11001100010101111000101 \\ \hline \text{C} & = & 10010110101111011000010 \end{array}$$

$$\text{C} = \text{M} \oplus \text{K} \quad (\text{bit-wise XOR})$$

For all i:     $\text{C}[i] = \text{M}[i] + \text{K}[i] \pmod{2}$

# One-time pad

**M** = message      **K** = key      **C** = encrypted message  
(everything in binary)

## Decryption:

$$\begin{array}{rcl} \text{C} = & 10010110101111011000010 \\ \oplus & \text{K} = & 11001100010101111000101 \\ \hline \text{M} = & 01011010111010100000111 \end{array}$$

Encryption:  $\text{C} = \text{M} \oplus \text{K}$

Decryption:  $\text{C} \oplus \text{K} = (\text{M} \oplus \text{K}) \oplus \text{K} = \text{M} \oplus (\text{K} \oplus \text{K}) = \text{M}$   
(because  $\text{K} \oplus \text{K} = 0$ )

# One-time pad

$$\begin{array}{rcl} M = & 01011010111010100000111 \\ \oplus K = & 11001100010101111000101 \\ \hline C = & 10010110101111011000010 \end{array}$$

One-time pad is perfectly secure:

For any  $M$ , if  $K$  is chosen uniformly at random, then  $C$  is uniformly at random.

So adversary learns nothing about  $M$  by seeing  $C$ .

But the shared key has to be as long as the message!  
Could we reuse the key?

# One-time pad

$$\begin{array}{r} M = 01011010111010100000111 \\ \oplus K = 11001100010101111000101 \\ \hline C = 10010110101111011000010 \end{array}$$

Could we reuse the key?

One-time only:

Suppose you encrypt two messages  $M_1$  and  $M_2$  with  $K$

$$C_1 = M_1 \oplus K$$

$$C_2 = M_2 \oplus K$$

$$\text{Then } C_1 \oplus C_2 = M_1 \oplus M_2$$

# Shannon's Theorem

Is it possible to have a secure system like one-time pad with a smaller key size?

Shannon proved “no”.

If  $K$  is shorter than  $M$ :

An adversary with **unlimited computational power** can learn some information about  $M$ .

# Question

What if we relax the assumption that the adversary is **computationally unbounded**?

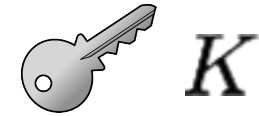
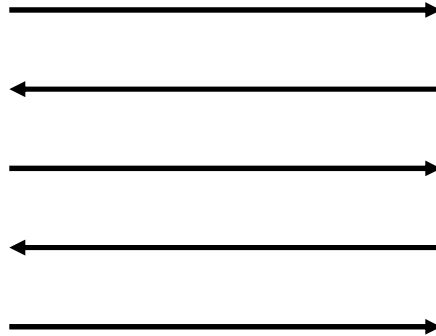
We can find a way to share a random secret key.  
(over an insecure channel)

We can get rid of the secret key sharing part.  
(**public key cryptography**)

# Secret Key Sharing



# Secret Key Sharing



# Diffie-Hellman key exchange

1976



**Whitfield Diffie**



**Martin Hellman**

# Diffie-Hellman key exchange

In  $\mathbb{Z}_N^*$

$$(B, E, N) \rightarrow \boxed{\text{EXP}} \rightarrow B^E \bmod N \quad \text{easy}$$

$$(B^E, B, N) \rightarrow \boxed{\text{LOG}_B} \rightarrow E \quad \text{seems hard}$$

Want to make sure for the inputs we pick, LOG is **hard**.

e.g. we don't want  $B^0 \ B^1 \ B^2 \ B^3 \ B^4 \ \dots$   
 $\quad \quad \quad \parallel \quad \parallel \quad \parallel \quad \parallel \quad \parallel$   
 $\quad \quad \quad 1 \quad B \quad 1 \quad B \quad 1 \quad \dots$

Much better to have a **generator**  $B$ .

# Diffie-Hellman key exchange

In  $\mathbb{Z}_N^*$

$$(B, E, N) \rightarrow \boxed{\text{EXP}} \rightarrow B^E \bmod N \quad \text{easy}$$

$$(B^E, B, N) \rightarrow \boxed{\text{LOG}_B} \rightarrow E \quad \begin{array}{l} \text{seems} \\ \text{hard} \end{array}$$

We'll pick  $N = P$  a prime number.

(This ensures there is a generator in  $\mathbb{Z}_P^*$  .)

We'll pick  $B \in \mathbb{Z}_P^*$  so that it is a **generator**.

$$\{B^0, B^1, B^2, B^3, \dots, B^{P-2}\} = \mathbb{Z}_P^*$$

# Diffie-Hellman key exchange



Pick prime  $P$

Pick generator  $B \in \mathbb{Z}_P^*$

Pick **random**  $E_1 \in \mathbb{Z}_{\varphi(P)}$

$\xrightarrow{P, B, B^{E_1}}$

$P, B, B^{E_1}$

Pick **random**  $E_2 \in \mathbb{Z}_{\varphi(P)}$

$\xleftarrow{B^{E_2}}$

Compute

$$(B^{E_2})^{E_1} = \boxed{B^{E_1 E_2}}$$

Compute

$$(B^{E_1})^{E_2} = \boxed{B^{E_1 E_2}}$$

# Diffie-Hellman key exchange



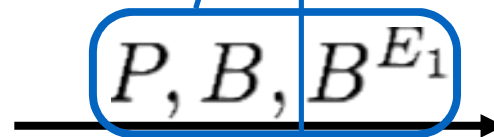
Pick prime  $P$

Pick generator  $B \in \mathbb{Z}_P^*$

Pick **random**  $E_1 \in \mathbb{Z}_{\varphi(P)}$

This is what the adversary sees.

If he can compute  $\text{LOG}_B$   
we are screwed!



$P, B, B^{E_1}$

Pick **random**  $E_2 \in \mathbb{Z}_{\varphi(P)}$



Compute

$$(B^{E_2})^{E_1} = \boxed{B^{E_1 E_2}}$$

Compute

$$(B^{E_1})^{E_2} = \boxed{B^{E_1 E_2}}$$

# Secure?

Adversary sees:  $P, B, B^{E_1}, B^{E_2}$

Hopefully he can't compute  $E_1$  from  $B^{E_1}$ .  
(our hope is that  $\text{LOG}_B$  is **hard**)

Good news: No one knows how to compute  $\text{LOG}_B$  efficiently.

Bad news: Proving that it cannot be computed efficiently is at least as hard as the P vs NP problem.

Diffie-Hellman assumption:

Computing  $B^{E_1 E_2}$  from  $P, B, B^{E_1}, B^{E_2}$  is hard.

Decisional Diffie-Hellman assumption:

You actually learn no information about  $B^{E_1 E_2}$

**One can use:**

**Diffie-Hellman  
(to share a secret key)**

**+**

**One-time Pad**

**for secure message transmissions**

**Note**

**This is as secure as its weakest link, i.e. Diffie-Hellman.**



# Question

What if we relax the assumption that the adversary is **computationally unbounded**?

We can find a way to share a random secret key.  
(over an insecure channel)

→ We can get rid of the secret key sharing part.  
(public key cryptography)

# **Public Key Cryptography**

# Public Key Cryptography



public



private

# Public Key Cryptography



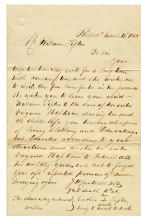
public



private

Can be used to lock.  
But can't be used to unlock.

# Public key cryptography



$M$

EMUPPZLBPAXYUDBIKZLDRNSHONTIJ  
YUQUXBMVUULITRVTQTKYRDMPP  
VPPUDHEWFFZYVDBSUTSPANCE  
GEMKBYDHGCTJGDMIAFTFAGALG  
TIMVZJANULVQKQADADVPFUNDUNA  
MGLLEDOYUULENFTJLACQVUDFVFA  
VIZETKZMYDIPFBIHJHKKWGLAEFTI  
BDDDDFTIWKHCTWQYDPTVQARE  
FLYETFEANALITVIBTFFKLLAIDF  
FLGUTETFRZBFDVYGGIPIFXHDKP  
FRQCTPLACUVOVMGCAQMSNEPFA  
ELEZYRQKFFVOREXDMVPNFQXELRE  
DNGRPFELPLPRLTALAGNUPOSTE  
DUMBERMDHAPADGNPLGFWLLAETG  
ENDYASQNLGMBGCEGDDYBREAL  
CTHREUJDELLEKASQVDBYBANE  
TPINQATUNHABERLONLEPLIACAE  
WNTWSDTEENACTESUDSTWRADE  
IFORSEDIWENHARIOITFAHENCEAICH  
LEESUSAGUVEUJZUOLMAEETETI  
ASEDNIAHTTMTWPIEROGAETWTFER  
ASTDDELLGSESTEGEASDDSTOLOST  
RELMLEHAGTDHADDVNOBMGPMFEURE  
ECONBETFEAMINISSTVNDWYQKES  
LUXUQKESOLIFRWEKQKESUQKES  
TWYQKESSEKZZWATKLDIAWYQKES  
VTTEZFQWYQKESZJQKESKUEAETGAE

$C$



$K_{\text{pub}}$



$K_{\text{pri}}$

$(M, K_{\text{pub}})$

Enc

$C$

Enc should be “one-way”.

Try to ensure it using  
computational  
complexity.

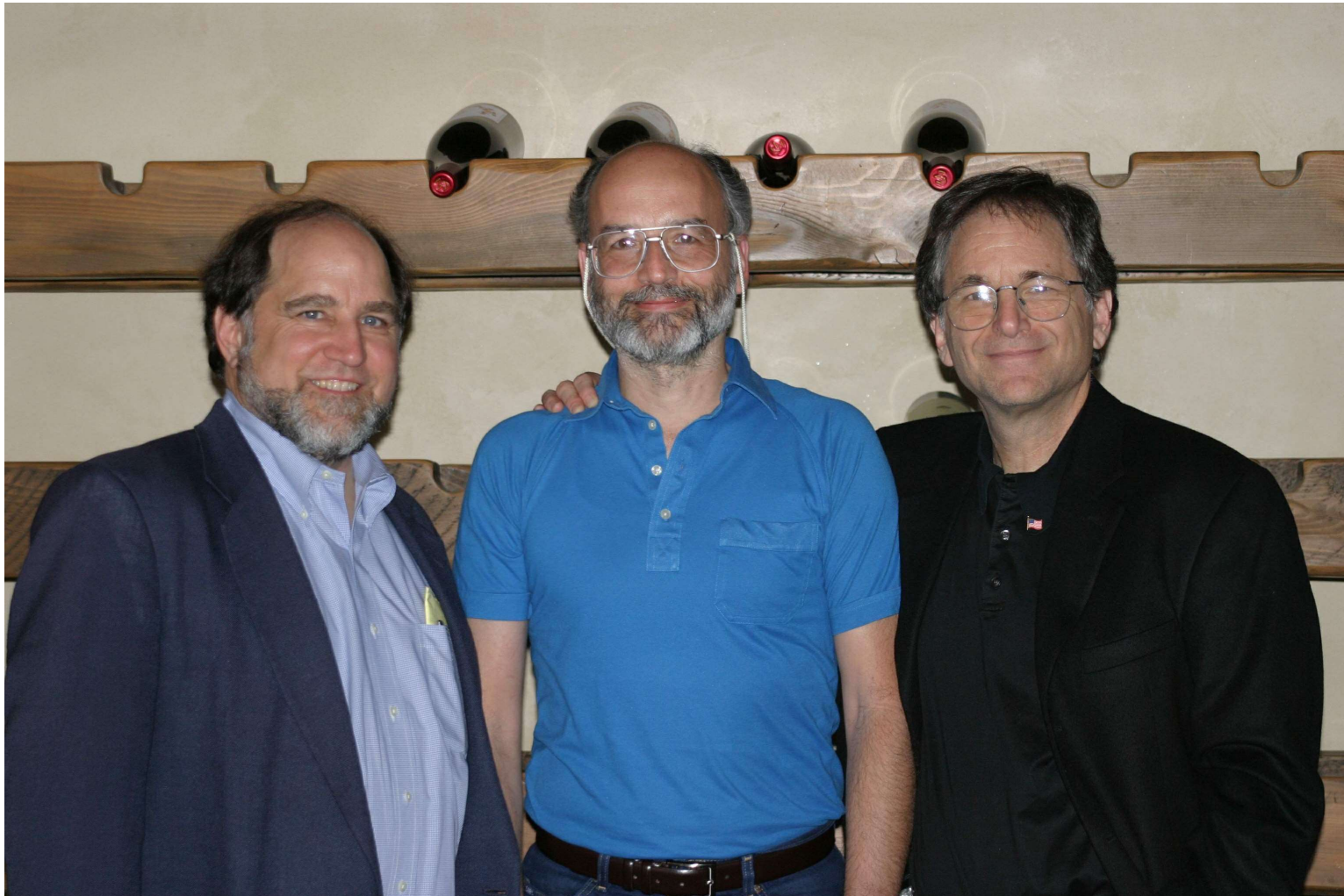
$(C, K_{\text{pri}})$

Dec

$M$

# RSA crypto system

1977



**Ron Rivest**

**Adi Shamir**

**Leonard Adleman**



# **RSA crypto system**



**Clifford Cocks**

**Discovered RSA system 3 years before them.**

**Remained secret until 1997. (classified information)**

# RSA crypto system

In  $\mathbb{Z}_N^*$

$$(B, E, N) \rightarrow \boxed{\text{EXP}} \rightarrow B^E \bmod N \quad \text{easy}$$

$$(B^E, E, N) \rightarrow \boxed{\text{ROOT}_E} \rightarrow B \quad \text{seems hard}$$

What if we encode using EXP ?

$$\text{assume } (M = B) \in \mathbb{Z}_N^*$$

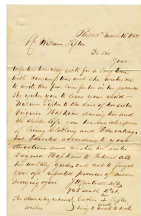
Public key can be  $(E, N)$  .

$$\text{and } E \in \mathbb{Z}_{\varphi(N)}$$

$$(M, K_{\text{pub}}) = (M, E, N) \rightarrow \boxed{\text{Enc}} \rightarrow M^E \bmod N = C$$



# RSA crypto system



$M$

EMUPPZLIFAXYIBDKZLKRNSHONTIV  
YOTUXBHVIVLITRVTOTWKTROMPO  
VPTDIBHEWTFZYQWREKETSAPNCE  
GEMKBYTHURCPADJGDMIAFPKAGLO  
TIMVZJANULVQKQADGVFPJUNOEUNA  
MGLLEDOYALLENFTJLACQVDFVFA  
YIEZKZMYDQFPRJHFWKQWGLAEFTI  
BDDDDFTOWKHTUWQYDPTQATSE  
FLYKTFEMOALITVIBTFTKLLAIDR  
FLGTETFRZBFDVVGQIPFPHDHRK  
FRQHTPLACUOVVIMOLAGUUSVFP  
ELEZYRONFFVORXDMVPNFQXELRE  
DNDKFPZELFCWJITALAGUUSVFP  
DUMBERDMDHAFJGNDPLGFWLLATG  
DNYASRQNLAMHCOFEDDYBREAL  
GTHREYUJLHLLKASRQNYBANE  
TPINGATHNHAHESLONLELPIAGAE  
WNTWSDTENDACTESUDSTWRADE  
ITPESDIWENHARIOITFAHENCEAICH  
LESUSAPUVEUJAZOALWALEFTETI  
ASPDNIAHTTSTWPIEROGAHTWFFER  
ASTDDELLSEISTEKEGASDDSTLOST  
RELMLEAGTSHADVNOBOMFMEURE  
ECONBTEFAMINSISTVNOBOMFMEURE  
LUXUONENOLIFWVLEQVATVNTD  
TWYOSSEKZKZATKLDIAWIRVENTD  
VTMEZFWODKZATJODHUEAVERGAE

$C$



$(N, E)$

$\cap$

$\mathbb{Z}_{\varphi(N)}$



$K_{\text{pri}}$

$(M, E, N)$

EXP

$$C = M^E \bmod N$$

Private key should allow  
us to invert EXP.

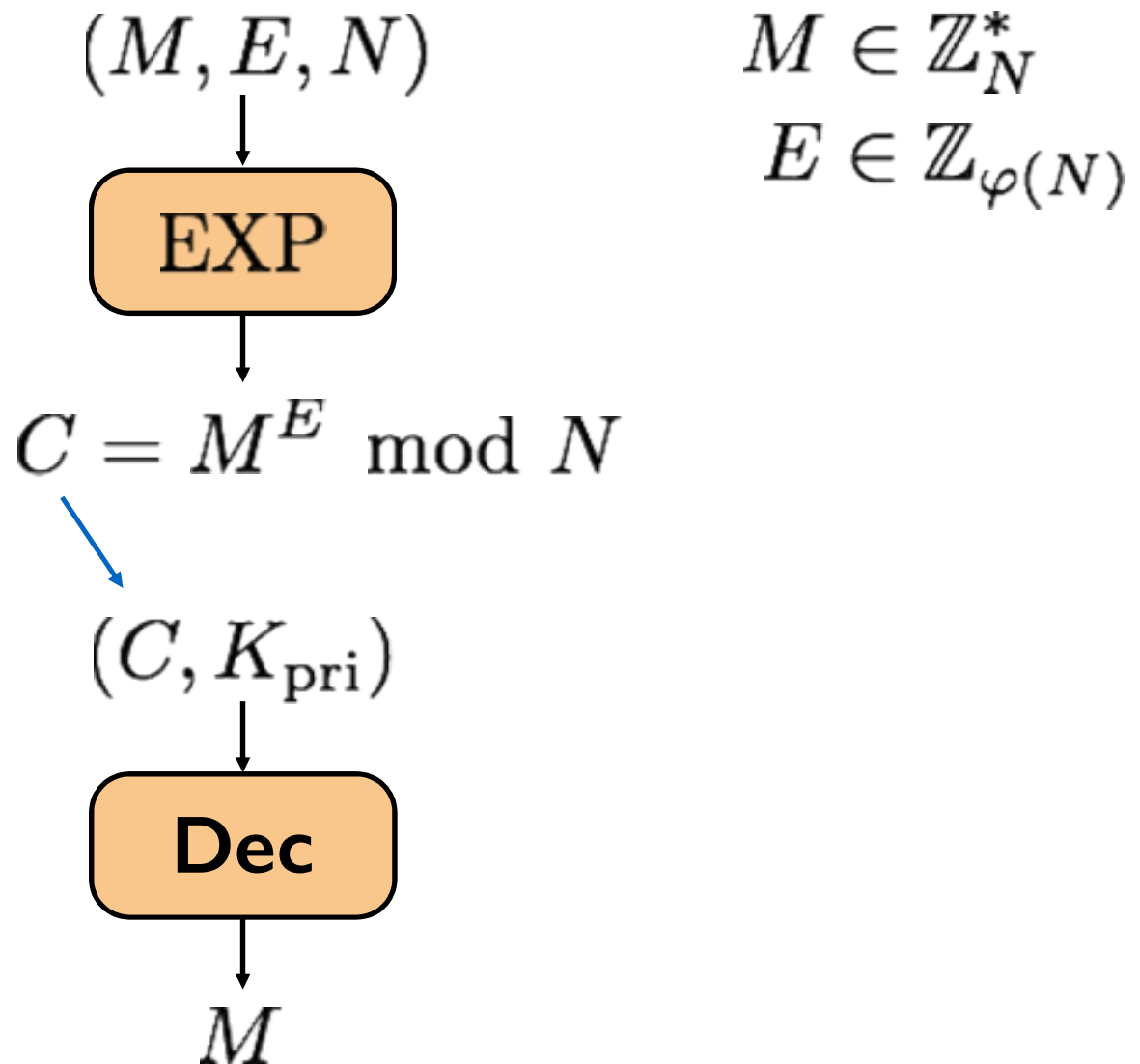
i.e. compute  $\text{ROOT}_E$

$(C, K_{\text{pri}})$

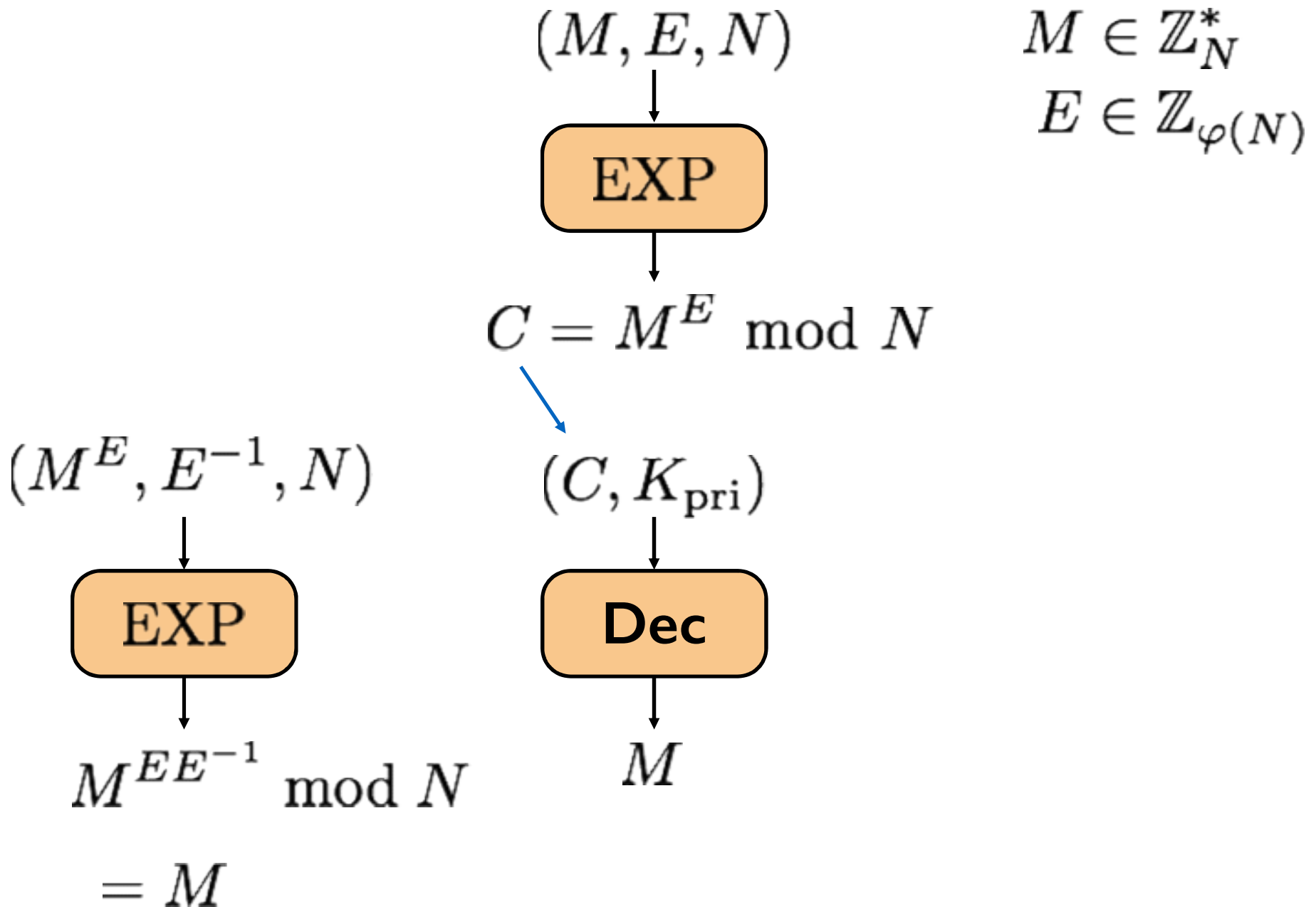
Dec

$M$

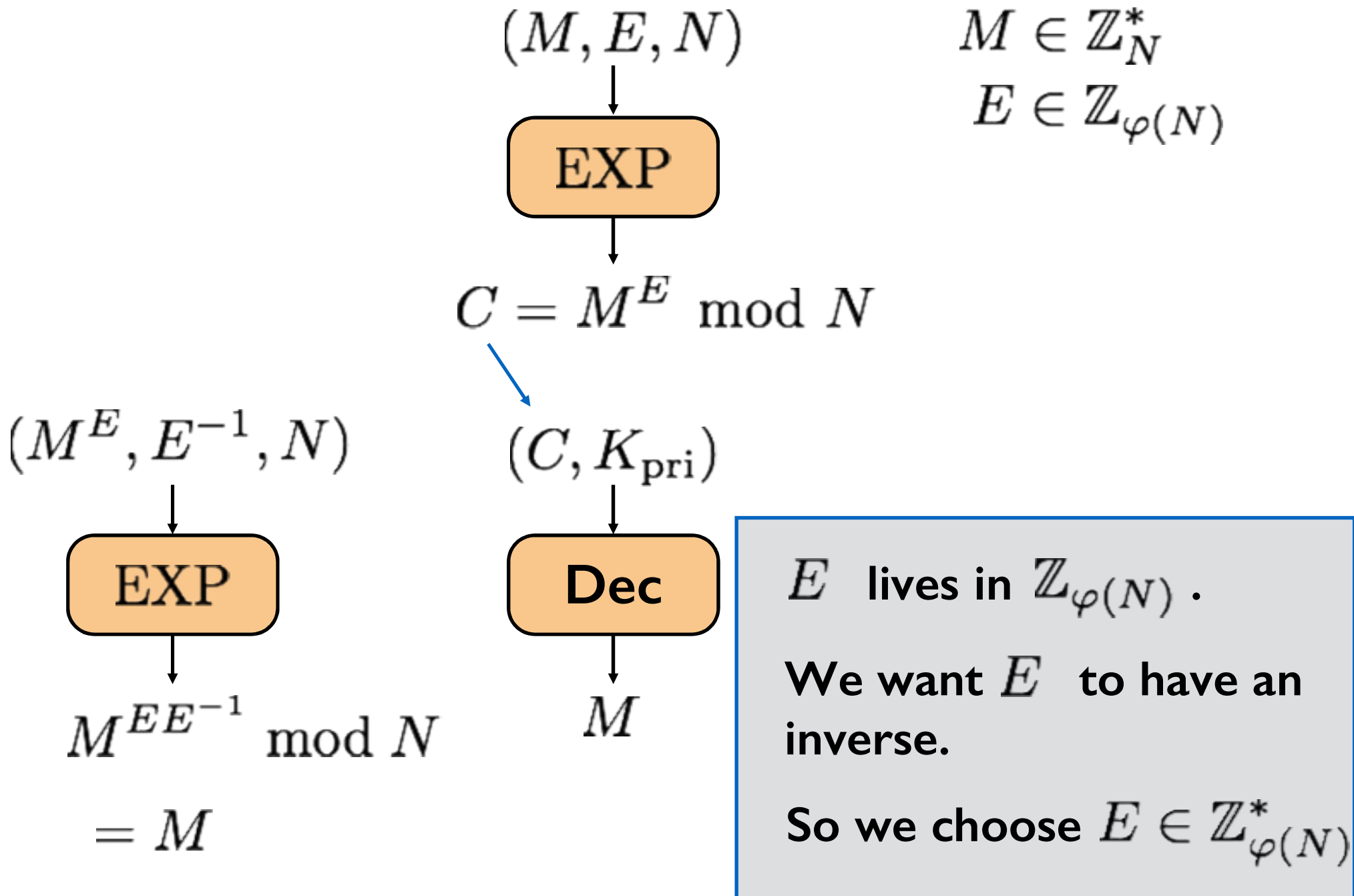
# RSA crypto system



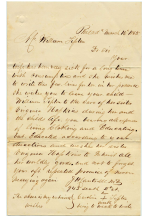
# RSA crypto system



# RSA crypto system



# RSA crypto system



$M$

EMUPPZLRFANYOSDJZLOKNNHONFIV  
VTFKIDHAWVTLITFENKJFTHESFHOSES  
VFTHUREHWEZLYVWNEKLTUJALNES  
GOWRKEFHMCFFQZDMHAGFPXNBRLG  
TMYEJZANUJQGDADVPFUDIEFVLA  
QGEZLCOYXUEENJTRILAGCTJLDFHBR  
VIEZEFENYOFKSEIPEWRECVWELFETI  
EVLDAFEXIOMAILTUUBVAPPEULAVIX  
PLASTETPFCBPTKXVHOPPTFONRKY  
FRANTQPUAECNUVPJMGCLAMUNEDPA  
ELEVENRDFVQZGDMWYONKQZLGBES  
DNEKRETFZLFLPMJLALWGRUNFOVLE  
DQUMERDMHDAFNUGKNUPLGEWILLAKTS  
EDOFAMONLANRGOOPTGORDYBENAJA  
CRTNREUTLOJLILKORNNORMWXMNE  
TPNGOJTHSALVPEUNLEKELPLACAS  
WMTWNOFRENBAHCTENSDRETNHABDE  
EPLASREIEMENRQIOTPLAKENCTALCS  
TEEPQARFOTUTUAEOTQARNAEERTNRTI  
REEDNGLAATTOSFWECSOAGREWFEN  
ACTDWHLCENITTEGQEAOSD BY DLOHT  
ECLMELMADAPADGKOVHOFQFQRE  
EYONREFAKREKALCEINJYJWYORKS  
DOXOHOJLSOLIFBFWFLBVAQPNGRSSO  
TWTSHASRUCZFWJLGLDIAWIFRFTY  
VTTMZFPRWDRZXTJOMKURUAUEKCAR

$C$



$(N, E)$



$N = PQ$   
 $E \in \mathbb{Z}_{\varphi(N)}^*$



$E^{-1}$

$(M, E, N)$

EXP

$$M^E = C$$

$(C, E^{-1}, N)$

EXP

$$M = C^{E^{-1}}$$

# RSA crypto system



EMUFPHZLFHXAYUDBJKZLDKSHNHFVIF  
TQTXQXHQVYVYVLFTRFVYQMYKDFHDF  
GQWKKHDFQMGCFQZBQMGIQFQXQIGL  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
MHDDDVHVDQWKBUPWPNVDFDYQZGZ  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
FLTFSTFSTFSTFSTFSTFSTFSTFSTF  
FLTFSTFSTFSTFSTFSTFSTFSTFSTF  
KLEZVHGFQFVQREXIKHVMVNPQXZL  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
QUMGEMDMHDAFJQZNPQWGLWJLAET  
TNDFARHONLRBHQOFQTHOIBDNRNFI  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
TPNATGATNRNAPENLNELNEPLACIA  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
TQFEDDITWENMIAQVYTRQHEQNTQA  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
TEQSFQFIOTUEQAEQVQAEAEAEAEAE  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
AECTDDHICHSIRTEQAEQASDDBYDF  
RKLMLEAGTADRDPEQMGHGFQDQH  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
QOXGHLQBSOLFQWFBFWLQVQAPNQF  
QZQZQZQZQZQZQZQZQZQZQZQZQZ  
VTMTFQWQDQZTQDQKQHQAEKQA



C

$$M$$
 $(N, E)$ 

- $N = PQ$
- $E \in \mathbb{Z}_{\varphi(N)}^*$

  $E^{-1}$

# Why is $N = PQ$

(product of distinct primes)?

$$(M, E, N)$$
$$(C, E^{-1}, N)$$

# EXP

EXP

$$M^E = C$$

$$M = C^{E-1}$$

# What if, say, $N = P$ ?

# How to choose N

How does Margaery compute  $E^{-1}$ ?

Computing  $E^{-1} \in \mathbb{Z}_{\varphi(N)}^*$  is easy if you know  $\varphi(N)$

She knows  $P$  and  $Q$ , so  $\varphi(PQ) = (P-1)(Q-1)$ .



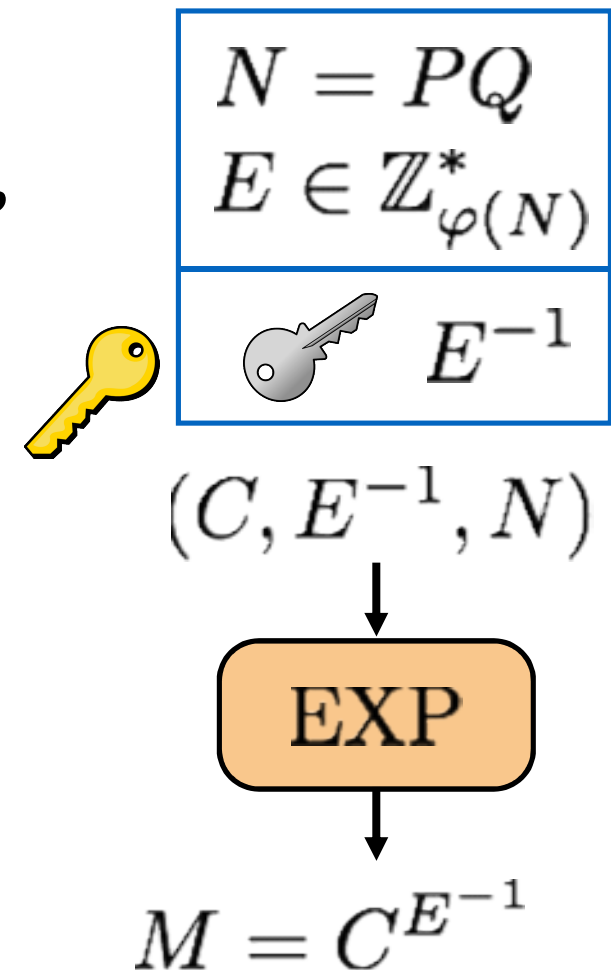
If the adversary can compute  $E^{-1} \in \mathbb{Z}_{\varphi(N)}^*$ ,  
we are screwed!

Adversary sees  $(N, E)$ .

Can he compute  $\varphi(N)$ ?

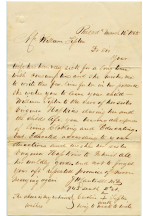
We believe this is computationally hard.

If the adversary can factor  $N$  efficiently,  
he can also compute  $\varphi(N)$ .





# RSA crypto system



$M$

EMUPPZLRFXYOSDJZLOKNNHONFIV  
VTFKIDHAWVTLTFRNCFHMSFPHMPS  
VTFKIDHAWVTLTFRNCFHMSFPHMPS  
GOWRKEFHMCFFZDMHAGFPXMBLQ  
TMYZLANUVOGDAQVFPDUEFVA  
QEGLECOYXUEENJTBLAGHTRJDFHBR  
VTEZEFYVDFKIDHAWVTLTFRNCFHMSF  
EVLDAFEXHOMBLTUGBYVFFEUJLAVIX  
PLASTETPFCBPFHAWHOPFUNDARY  
FRANTGPAAECHUVPDMMGLADMUHEDPA  
ELEVENHDFVQZDMHAWVTLTFRNCFHMSF  
DNEHREPLAFHMLALWGRUNFOVLES  
DQUMERDMHDAFNUGKNUPLGEWILLAKTS  
EDYFAMONLANHGOPTGORDYBENAJA  
CRTNRETVLQJLNLKORNNHOMWXMNE  
TPNGOHSRAVPSUNLEKELPLACAS  
WMTWNOFRENHACHENHUMETNHADE  
EPLASREPMENHAIQVPLASREPMENHAIQ  
TEEPQAFPTOTUTUAEOTAHMADEERTUTI  
REEDONLAAITDSEFWCEBOAGREWFEN  
ACTDWHLCENHETEGEASDDBYDLOHT  
ECLMELMADAFADGKORHOFGEHRE  
EYONHREPLAFHMLALWGRUNFOVLES  
DOXOHOHLSOLIFBFWFLBVAQPNHGRSSO  
TWTSHASREKFEWJLGLDIAWIFRFTY  
VTTMEFPKWDRZXTJOMKURUAUEKCAR

$C$



$(N, E)$



$N = PQ$   
 $E \in \mathbb{Z}_{\varphi(N)}^*$



$E^{-1}$

$(M, E, N)$

EXP

$$M^E = C$$

$(C, E^{-1}, N)$

EXP

$$M = C^{E^{-1}}$$



# Secure?

The advantage Margaery has over the adversary is that she can compute  $\varphi(N)$ .  
(and therefore  $E^{-1}$ )



$$N = PQ$$
$$E \in \mathbb{Z}_{\varphi(N)}^*$$



$$E^{-1}$$

If the adversary can factor  $N$  efficiently,  
he can also compute  $\varphi(N)$ .  
(and therefore  $E^{-1}$ )

$$(C, E^{-1}, N)$$



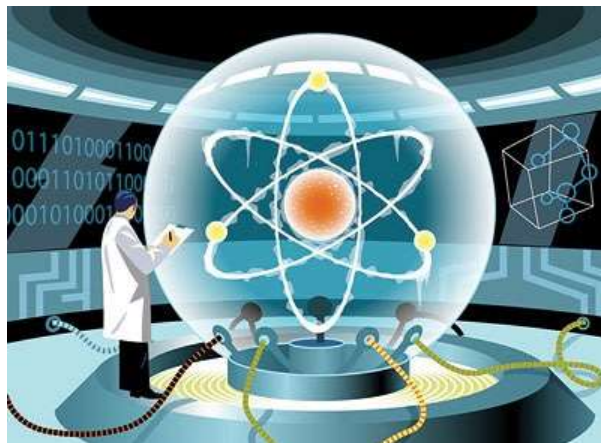
$$M = C^{E^{-1}}$$

# Concluding remarks

A variant of this is widely used in practice.

From  $N$ , if we can efficiently compute  $\varphi(N)$ , we can crack RSA.

If we can factor  $N$ , we can compute  $\varphi(N)$ .



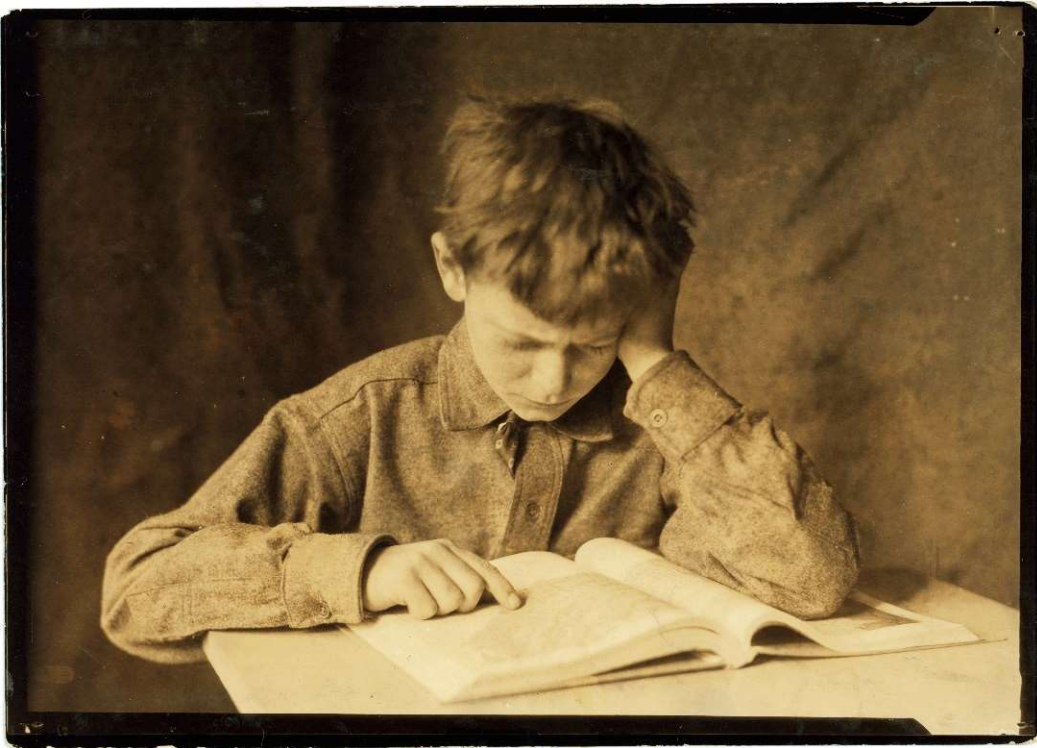
Quantum computers  
can factor efficiently.

Is this the only way to crack RSA?

We don't know!

So we are really hoping it is secure.

# Study Guide



## Modular Arithmetic:

- fast exponentiation
- generators
- hardness of root and logarithm (mod  $n$ )
- exp as a one-way func.

## Cryptographic Algorithms:

- Cesar Cypher
- One Time Pad
- Diffie Hellman  
(Secure Key Exchange)
- RSA  
(Public Key Encryption)