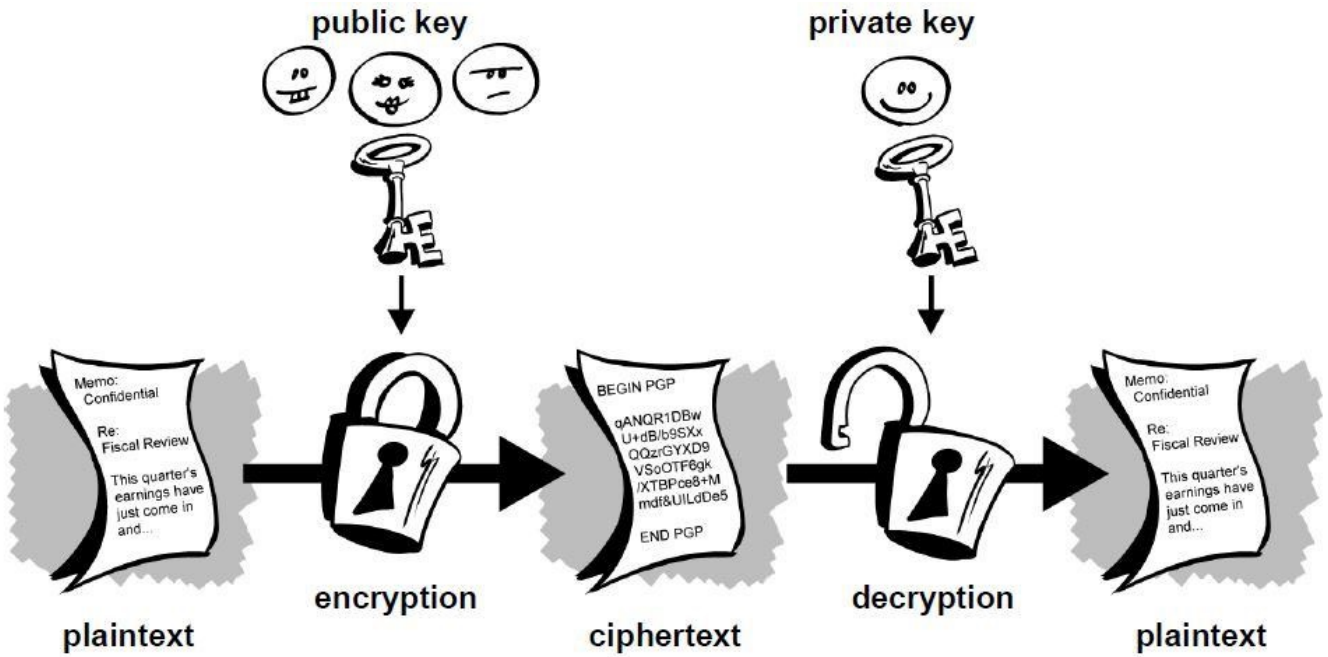
15-251
Great Theoretical Ideas in Computer Science

Lecture 22:
Cryptography

November 12th, 2015



What is cryptography about?

Adversary
Eavesdropper

"I will cut your throat"

"I will cut your throat"

What is cryptography about?

“loru23n8uladjkfb!#@”

“I will cut your throat”

encryption

“loru23n8uladjkfb!#@”

“loru23n8uladjkfb!#@”

decryption

“I will cut your throat”

What is cryptography about?

Study of protocols that avoid the bad affects of adversaries.

- Can we have secure online voting schemes?

- Can we use digital signatures.

- Can we do computation on encrypted data?

- 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, computational complexity revolutionized crypto.

Applications of computationally hard problems.

There is good math (e.g. number theory).

The plan

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

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

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

Review of Modular Arithmetic

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

Example

$N = 5$

0	1	2	3	4	5	6	7	8	9	10	11	12	$\cdots$
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	$\bmod 5$
0	1	2	3	4	0	1	2	3	4	0	1	2	$\cdots$

$\mathbb{Z}_5$ (= 0, 1, 2, 3, 4)

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

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

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

> 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 $A^B \bmod N$

Fast modular exponentiation: repeatedly square and mod.

> taking roots

> logarithm

→ No known efficient algorithm exists.

> 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 mult

> division A/B

Find B^{-1}. The

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

A modification of Euclid's Algorithm gives you B^{-1}**.**

> exponentiatio

Fast modular exponentiation. Repeatedly square and mod.

> taking roots

> logarithm

No known efficient algorithm exists.

$\mathbb{Z}_4$

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

$\mathbb{Z}_8^*$

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

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

behaves nicely with respect to *addition*

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

behaves nicely with respect to *multiplication*

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

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

if P, Q 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	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

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

$\varphi(5) = 4$

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

$$\forall A, \quad A^4 = 1 \quad \Longrightarrow A^{4k} = (A^4)^k = 1$$

Euler's Theorem:

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

Fermat's Little Theorem:

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

$$
\begin{array}{ccccc}
1 & & & & \\
\| & & & & \\
A^0 & A^1 & A^2 & \cdots & A^{\varphi(N)-1} \\
\| & \| & \| & & \| \\
A^{\varphi(N)} & A^{\varphi(N)+1} & A^{\varphi(N)+2} & \cdots & A^{2\varphi(N)-1} \\
\| & \| & \| & & \| \\
A^{2\varphi(N)} & A^{2\varphi(N)+1} & A^{2\varphi(N)+2} & \cdots & A^{3\varphi(N)-1}
\end{array}
$$

IMPORTANT

When exponentiating elements $A \in \mathbb{Z}_N^*$,

can think of the exponent living in the universe $\mathbb{Z}_{\varphi(N)}$.

In $\mathbb{Z}$

$(B, E) \rightarrow \boxed{\mathrm{EXP}} \rightarrow B^E$ **hard**

Two inverse functions:

$(B^E, E) \rightarrow \boxed{\mathrm{ROOT}_E} \rightarrow B$ **easy**

$(B^E, B) \rightarrow \boxed{\mathrm{LOG}_B} \rightarrow E$ **easy**

In $\mathbb{Z}$

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

easy

(1881676371789154860897069, 3) ⟶ 123456789

(can do binary search)

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

easy

(48519278097689642681155855396759336072749841943521979872827, 3)

⟶ 123

(keep dividing by B)

In $\mathbb{Z}_N^*$

$(B, E, N) \to$ EXP $\to B^E \mod N$ **easy**

Two inverse functions:

$(B^E, E, N) \to \text{ROOT}_E \to B$ **seems hard**

$(B^E, B, N) \to \text{LOG}_B \to E$ **seems hard**

Exercise: Convince yourself that the algorithms in the setting of $\mathbb{Z}$ do not work in $\mathbb{Z}_N^*$.

In $\mathbb{Z}_N^*$

$(B, E, N) \rightarrow$ EXP $\rightarrow B^E \mod N$ **easy**

Two inverse functions:

$(B^E, E, N) \rightarrow$ ROOT_E $\rightarrow B$ **seems hard**

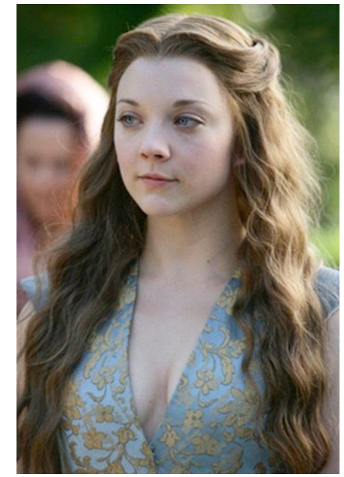
$(B^E, B, N) \rightarrow$ LOG_B $\rightarrow E$ **seems hard**

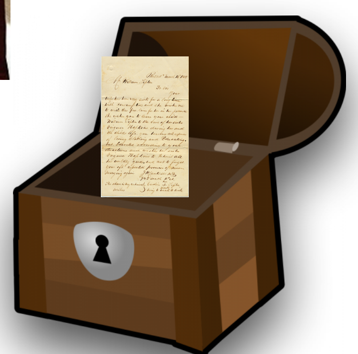
One-way function: easy to compute, hard to invert.

EXP seems to be one-way.

Private Key Cryptography

Private key cryptography



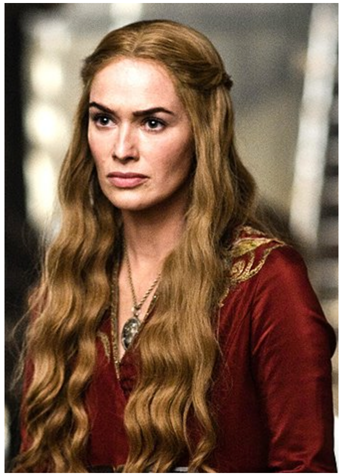


Parties must agree on a key pair beforehand.

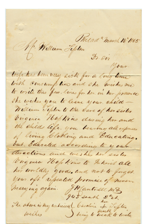
Private key cryptography

there must be a secure way of exchanging the key

Private key cryptography



C

 K_A  M (plaintext)

 K_B

(M, K_A) → Enc → C (ciphertext)

Enc should be "one-way".

Try to ensure it using the secrecy of the key.

(C, K_B) → Dec → M

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

<u>Example</u>: 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, sohdvh jurz xs dqg vroyh brxu rzq sureohpv."

 : the shift number

Easy to break.

Substitution cipher

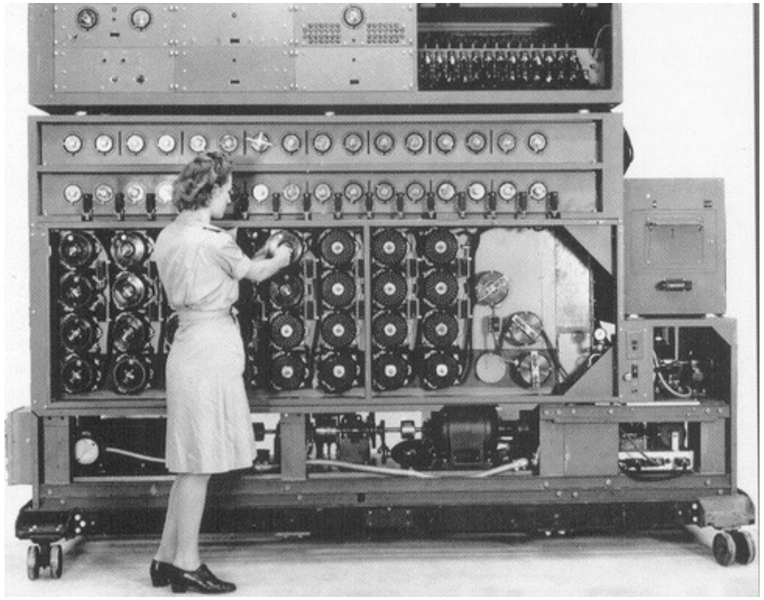
```
a b c d e f g h i j k l m n o p q r s t u v w x y z
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
j k b d e l m c f g n o x y r s v w z a t u p q h i
```

 : permutation of the alphabet

Easy to break by looking at letter frequencies.

Enigma

A much more complex cipher.



One-time pad

M = message K = key C = encrypted message

(everything in binary)

Encryption:

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

$C = M \oplus K$ (bit-wise XOR)

<u>For all i:</u> $C[i] = M[i] + K[i] \pmod 2$

One-time pad

M = message   K = key   C = encrypted message

(everything in binary)

Decryption:

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

<u>Encryption:</u> $C = M \oplus K$

<u>Decryption:</u> $C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M$

(because $K \oplus K = 0$)

One-time pad

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

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 you need to share a key that is as long as the message!

Could we reuse the key?

One-time pad

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

Could we reuse the key?

<u>One-time only:</u>

Suppose you encrypt two messages M_1 and M_2 with K.

$C_1 = M_1 \oplus K$

$C_2 = M_2 \oplus K$

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 could learn some information about M.

Question

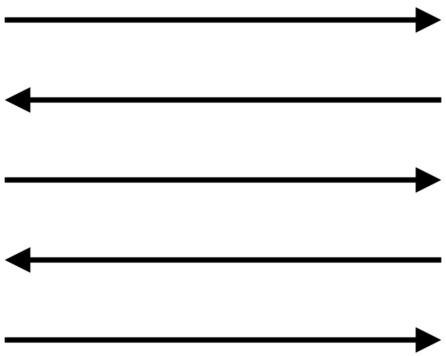
What if we relax the assumption that the adversary is computationally unbounded?

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



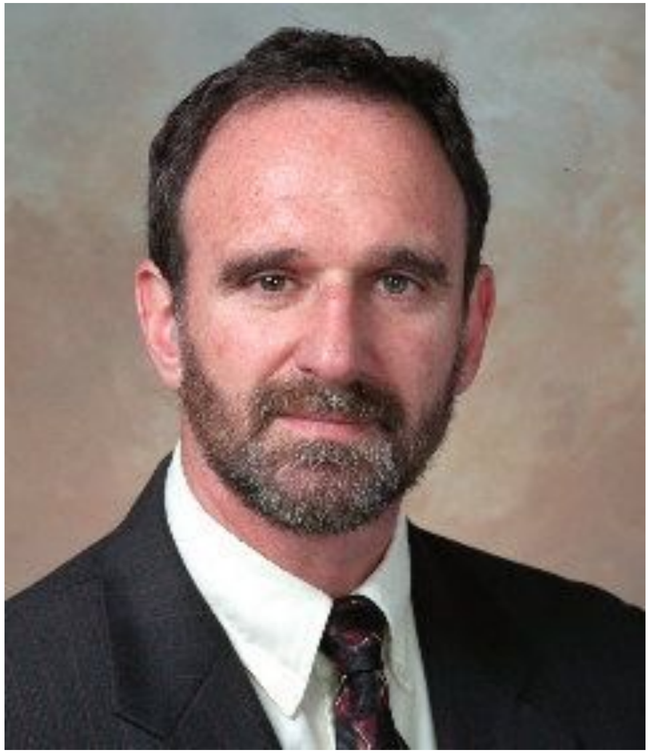
 K

 K

Diffie-Hellman key exchange

1976

Whitfield Diffie



Martin Hellman

Diffie-Hellman key exchange

In $\mathbb{Z}_N^*$

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

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

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

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

$$\begin{matrix} B^0 & B^1 & B^2 & B^3 & B^4 & \ldots \\ \| & \| & \| & \| & \| & \\ 1 & B & 1 & B & 1 & \ldots \end{matrix}$$

Much better to have a *generator* B.

Diffie-Hellman key exchange

In $\mathbb{Z}_N^*$

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

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

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*.

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

Diffie-Hellman key exchange

Pick prime P

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

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

P, B, B^{E_1} →

P, B, B^{E_1}

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

← B^{E_2}

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 LOG_B we are screwed!

P, B, B^{E_1} →

P, B, B^{E_1}

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

← B^{E_2}

B^{E_2}

Compute

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

Compute

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

Secure?

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

Hopefully he can't compute E_1 from B^{E_1}.
(our hope that LOG_B is hard)

Good news: No one knows how to compute 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 could use:

Diffie-Hellman
(to share a secret key)

+

One-time Pad

Note

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

Question

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

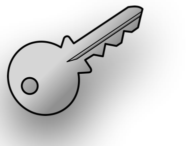
We can find a way to share a random secret key.
(over an insecure channel)

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

Public Key Cryptography

Public Key Cryptography

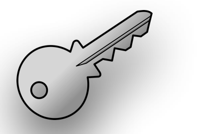
public



private

Public Key Cryptography

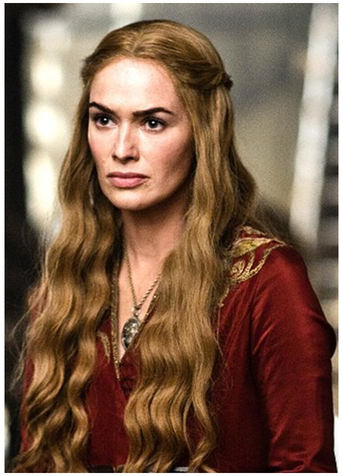
public



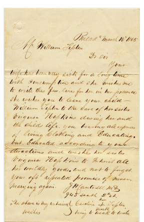
private

Can be used to lock.
But **<u>can't</u>** be used to unlock.

Public key cryptography



C



M

K_{pub}

K_{pri}

(M, K_{pub})

Enc

C

Enc should be "one-way".

Try to ensure it using computational complexity.

(C, K_{pri})

Dec

M

RSA crypto system

In $\mathbb{Z}_N^*$

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

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

What if we encode using EXP? $(M = B)$

Public key can be (E, N) .

$$(M, K_{\text{pub}}) = (M, E, N) \longrightarrow \boxed{\textsf{Enc}} \longrightarrow M^E \mod N$$

$$= C$$

RSA crypto system

In $\mathbb{Z}_N^*$

$(B, E, N) \rightarrow$ EXP $\rightarrow B^E \mod N$ **easy**

$(B^E, E, N) \rightarrow \mathrm{ROOT}_E \rightarrow B$ **seems hard**

What if we encode using EXP?

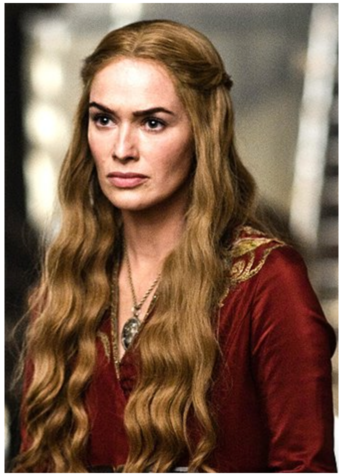
assume
$(M = B) \in \mathbb{Z}_N^*$

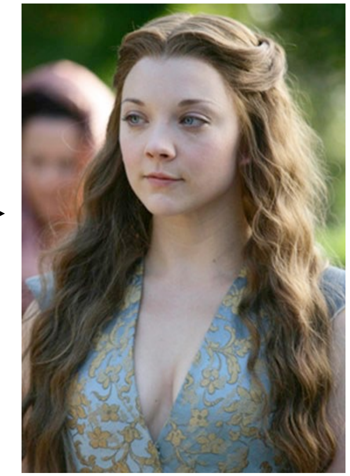
Public key can be (E, N) .

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

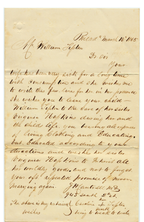
$(M, K_{\mathrm{pub}}) = (M, E, N) \rightarrow$ Enc $\rightarrow M^E \mod N$
$= C$

RSA crypto system





C



M

(N, E)

$\in$

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

K_{pri}

(M, E, N)

↓

EXP

↓

$C = M^E \mod N$

Private key should allow us to invert EXP.

i.e. compute ROOT_E

(C, K_{pri})

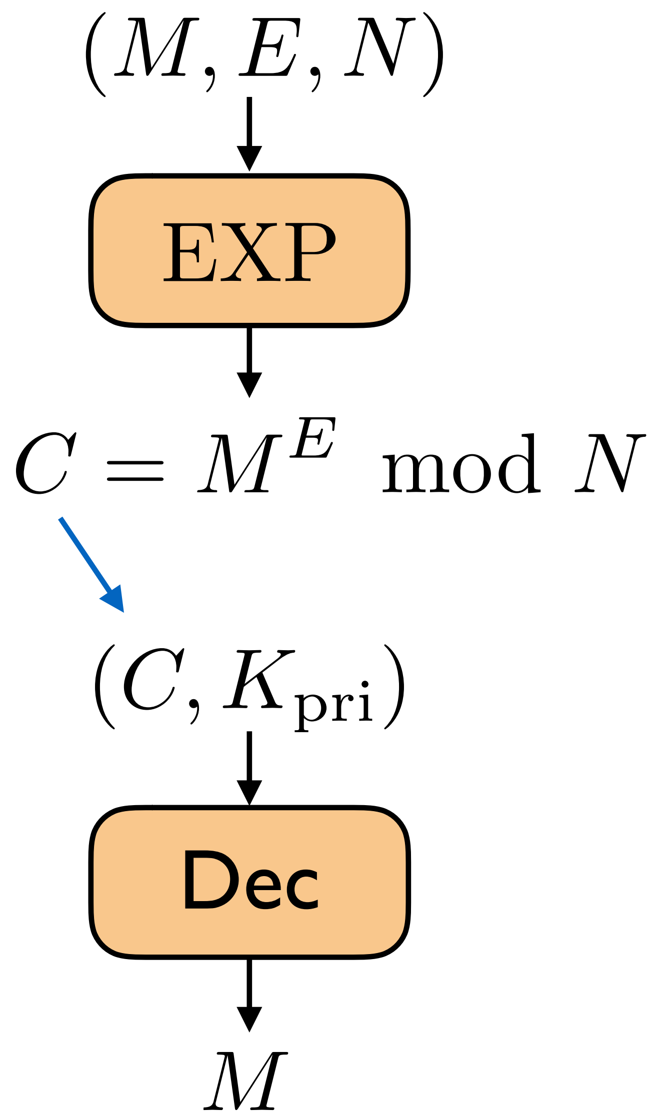
↓

Dec

↓

M

RSA crypto system



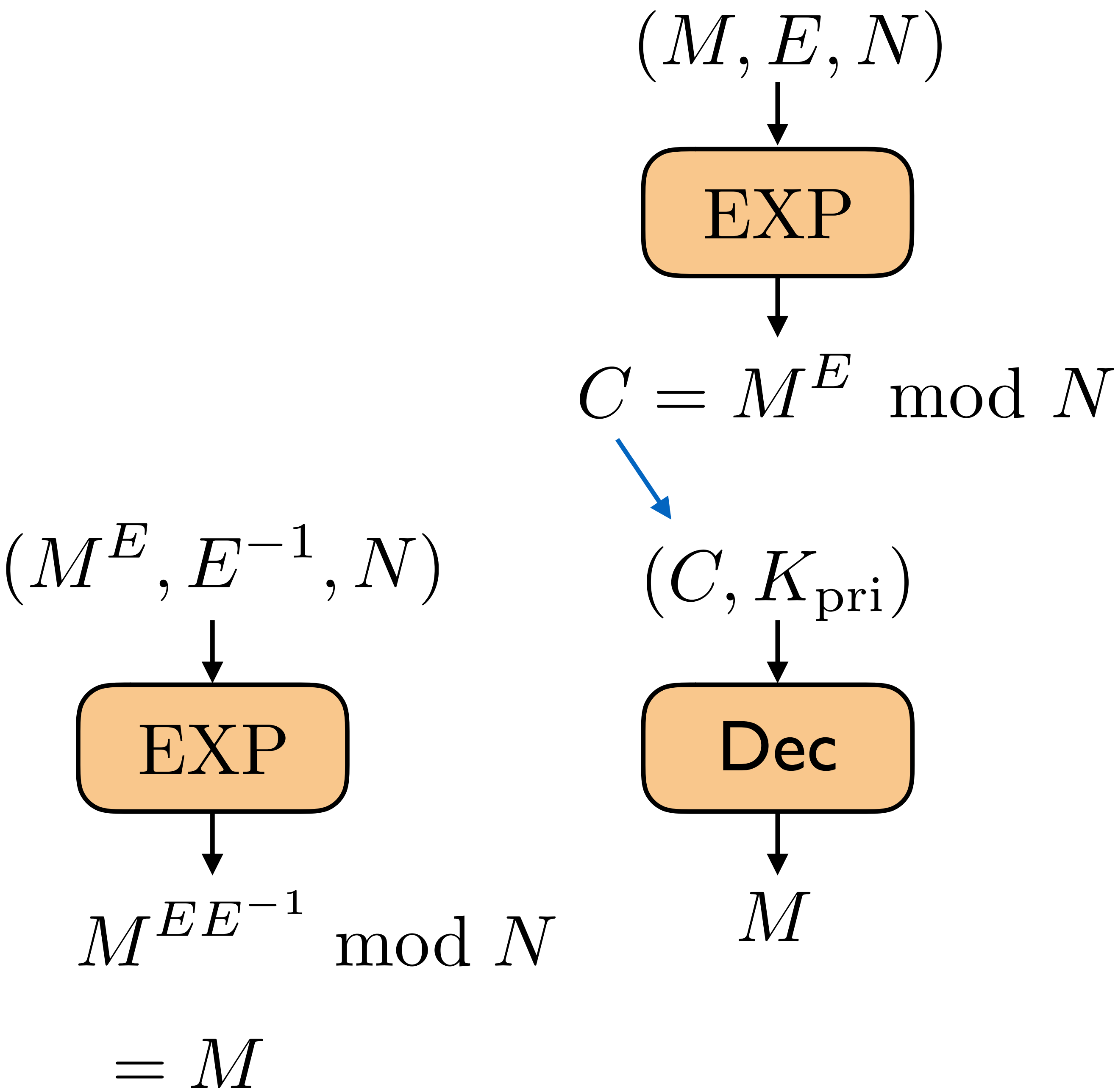
$$M \in \mathbb{Z}_N^*$$

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

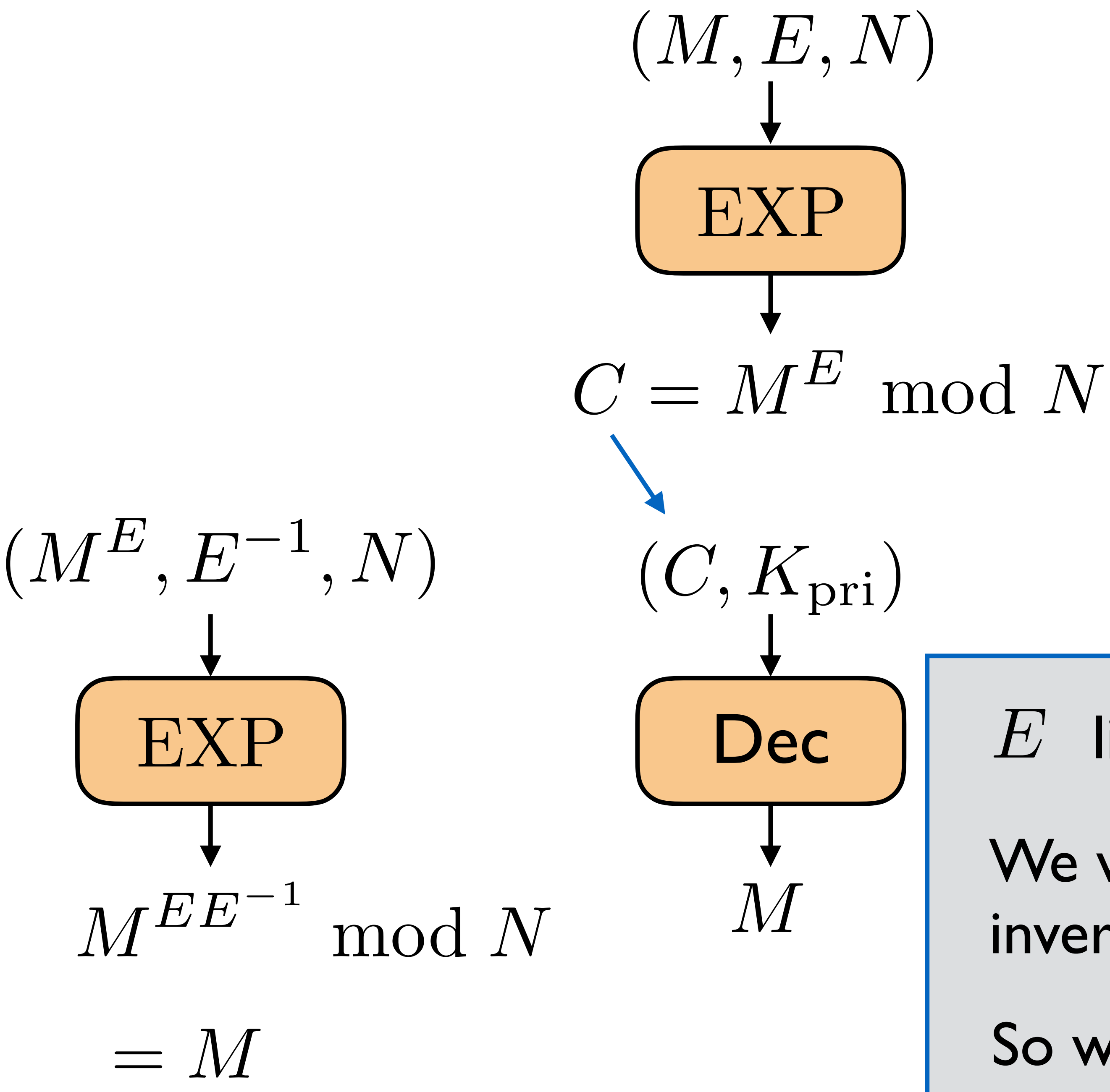
RSA crypto system

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

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



RSA crypto system



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

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

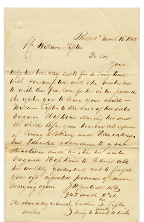
E lives in $\mathbb{Z}_{\varphi(N)}$.

We want E to have an inverse.

So we choose $E \in \mathbb{Z}_{\varphi(N)}^*$

RSA crypto system

C

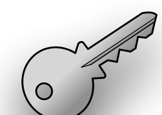


M

(N, E)

$N = PQ$

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



E^{-1}

(M, E, N)

EXP

$M^E = C$

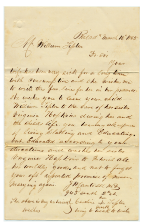
(C, E^{-1}, N)

EXP

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

RSA crypto system

C



M

(N, E)

$N = PQ$

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

E^{-1}

(M, E, N)

EXP

$M^E = C$

Why is *N* = *PQ*
(product of distinct primes)?

What if, say, *N* = *P* ?

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

EXP

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

Secure?

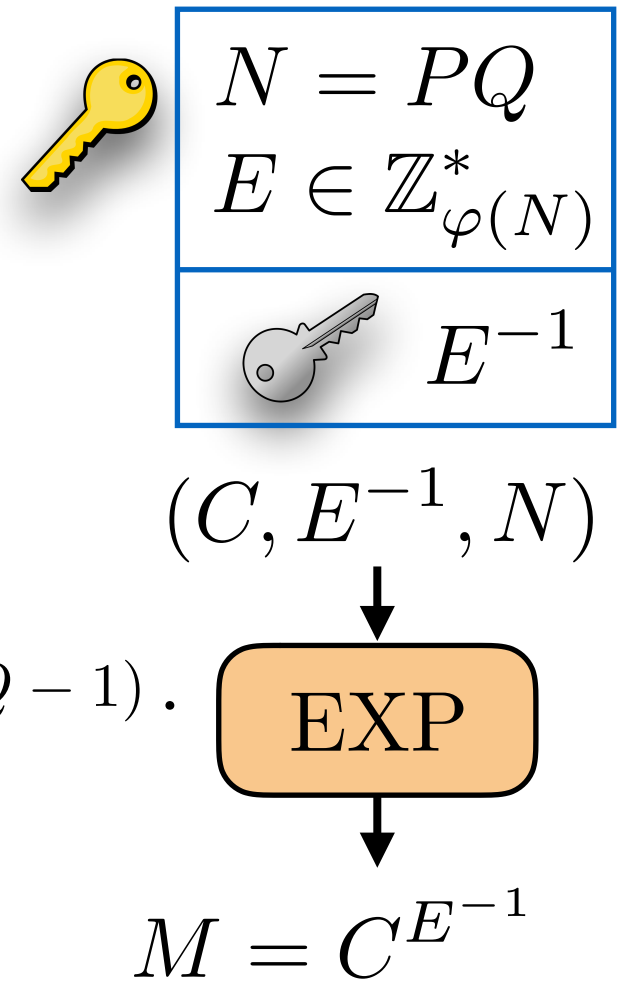
If the adversary can compute $E^{-1} \in \mathbb{Z}^*_{\varphi(N)}$,
we are screwed!

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

Adversary sees (N, E).
Can he compute $\varphi(N)$?
We believe this is computationally hard.

How does Margaery compute $\varphi(N)$?
She knows P and Q, so $\varphi(PQ) = (P-1)(Q-1)$.

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



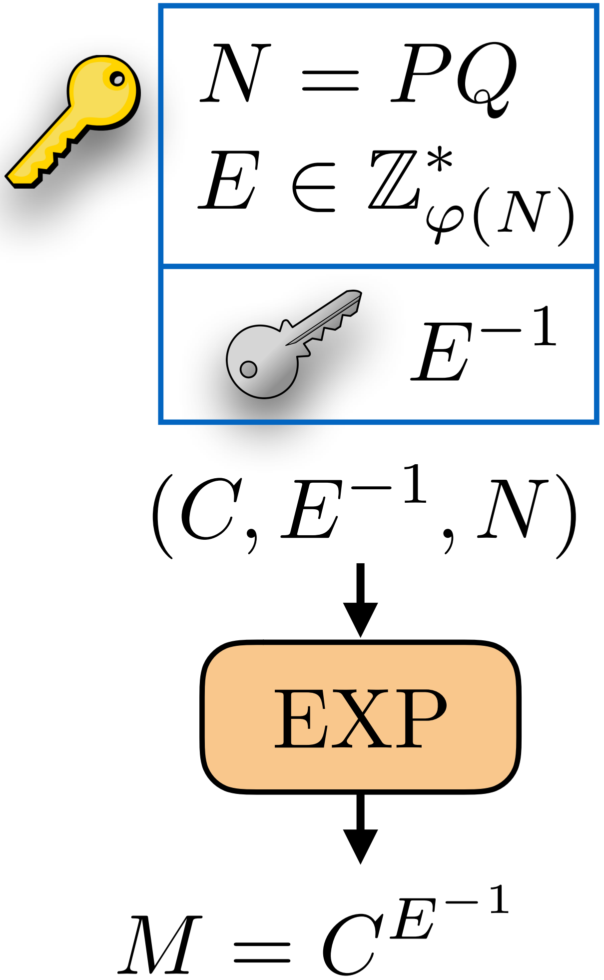
Secure?

The advantage Margaery has over the adversary is that she can compute $\varphi(N)$.

(and therefore E^{-1})

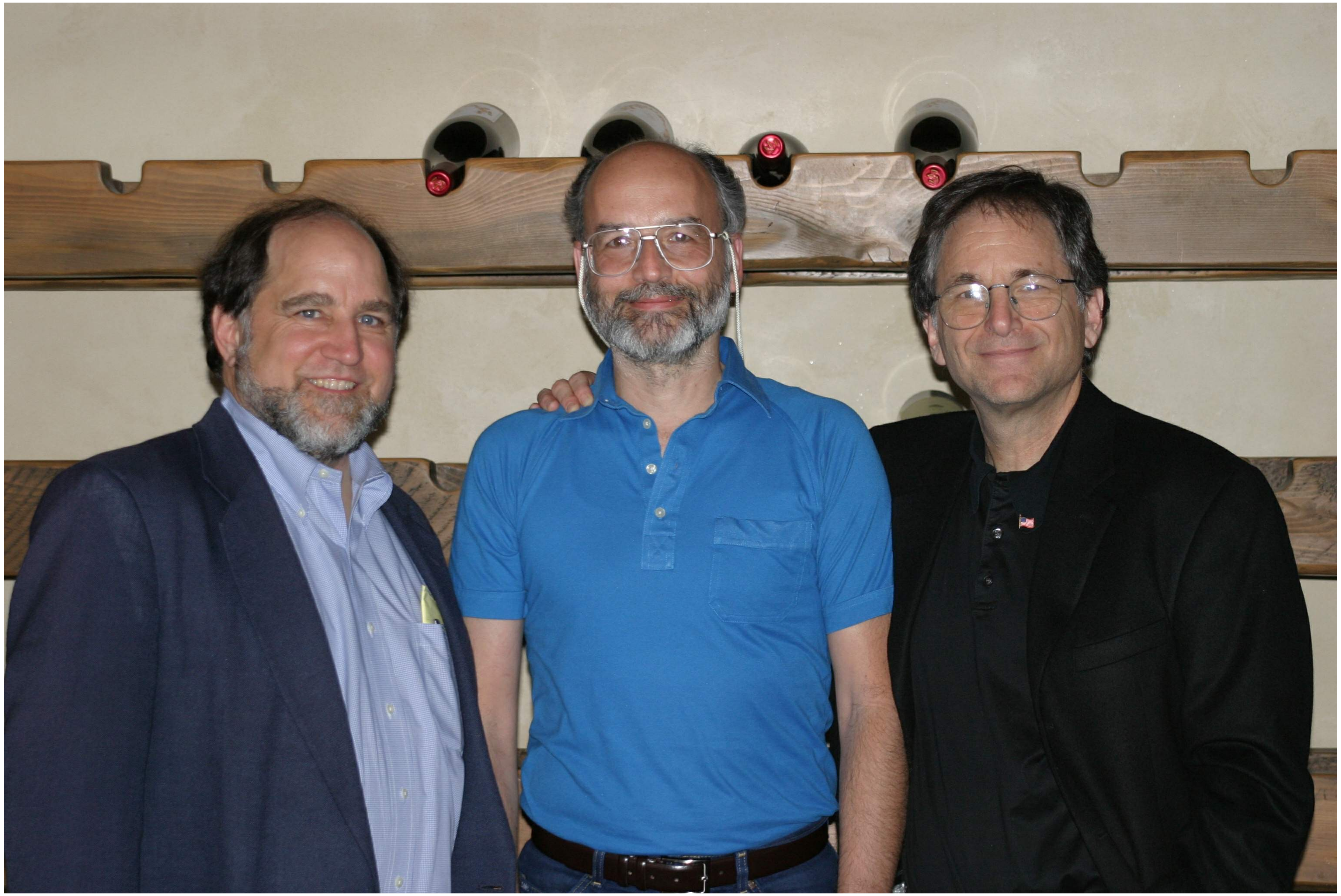
If the adversary can factor N efficiently, he can also compute $\varphi(N)$.

(and therefore E^{-1})



RSA crypto system

1977



Ron Rivest Adi Shamir Leonard Adleman

RSA crypto system

Clifford Cocks

Discovered RSA system 3 years before them.

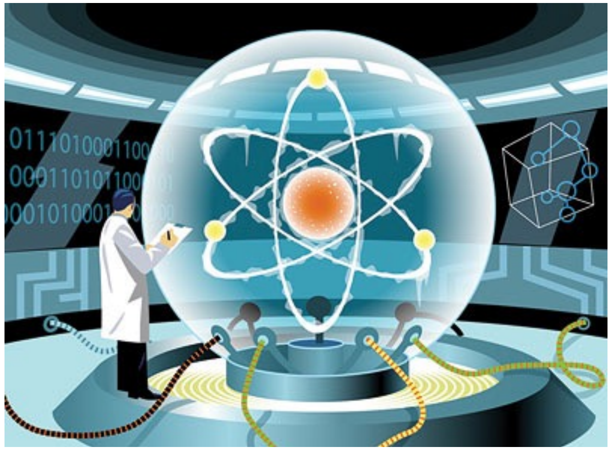
Remained secret until 1997. (was classified information)

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 <u>hoping</u> it is secure.